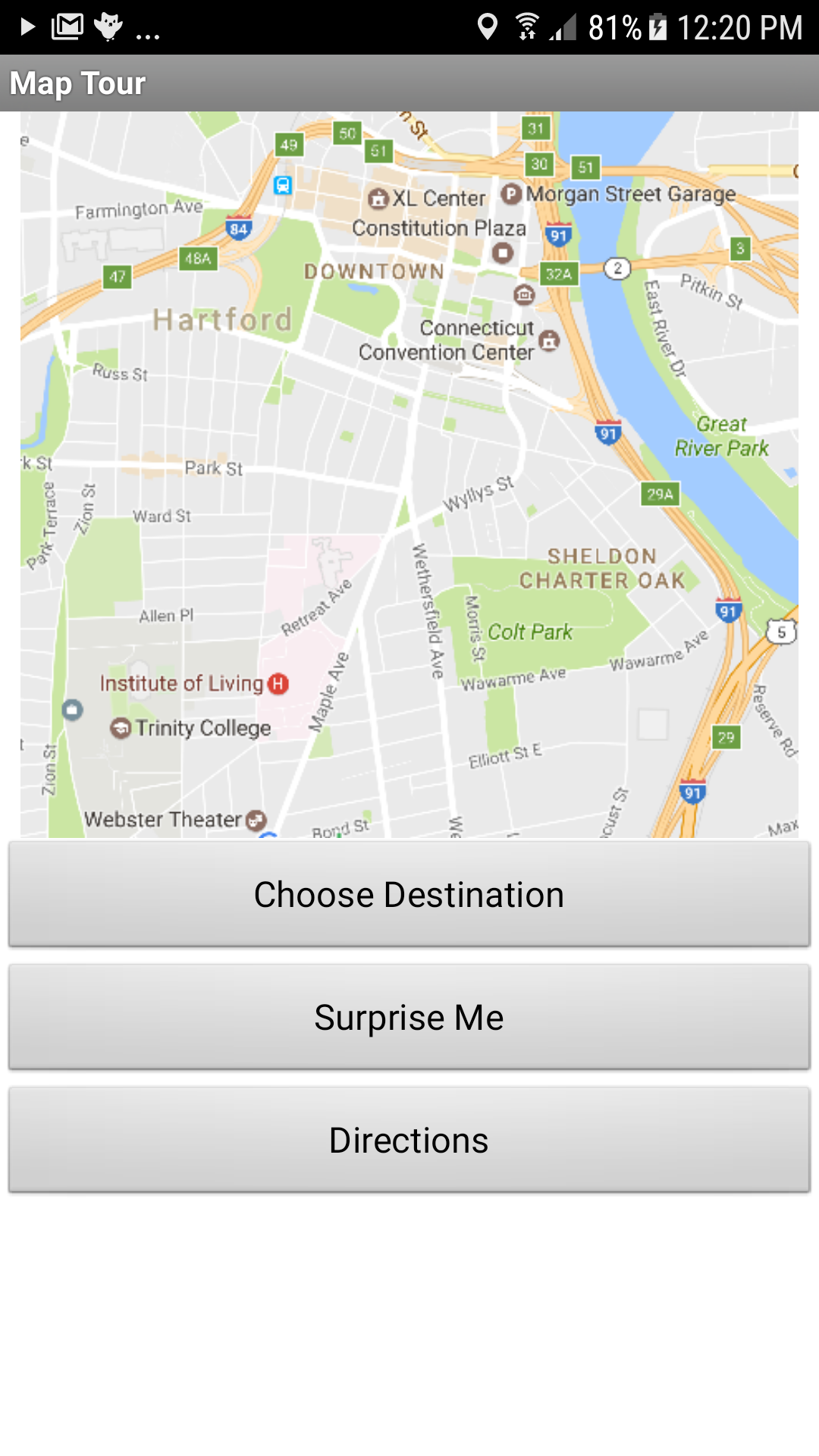
|  |  |
| --- | --- |
| The Map Tour allows a user to select a location from a list and it then launches the device’s Google Maps app to show the selected location on the map.    The app uses what’s known as the ***Google Maps Application Programming Interface (API)*** to enable the app to provide various forms of help and assistance such as:     * Finding a location on a map. * Getting directions (driving or walking) to a destination. * Finding restaurants or other places of interest on a map.     Thus, by learning to use the maps API this lesson opens up a whole new view of the Internet that is only available to programmers.  **Objectives:** In this lesson you will create an app that:   * uses the Google Maps API to perform different map-related functions; * uses a List to store and access data; * randomly selects items from a list; | ScreenShot.png  ***[Click to watch Preview Video](http://www.youtube.com/watch?v=fz8dOtSsdOY)*** |

## 

## Getting Ready

To get started, [open App Inventor with the Map Tour Template](http://ai2.appinventor.mit.edu/?repo=templates.appinventor.mit.edu/trincoll/csp/unit3/templates/MapTour2017/MapTourTemplate.asc) in a separate tab. When the project opens, use the *Save As* option to rename it ***MapTour***. Then follow along with the following tutorial.

## The Map Tour UI



The UI for the Map Tour app is very simple. It contains: an Image (a map of Hartford, CT, home of Trinity College), a *List Picker* (Choose Destination), two buttons (*Surprise Me* and *Directions*) and an *Activity Starter.* The Image is used to display a map of Hartford, Connecticut. Later on you can customize the UI with a map of your own neighborhood.

The *List Picker* is used to let the user choose a location that they would like to visit. Once the user chooses a destination, the *Activity Starter* opens the Google Maps app and displays that location on a map.

**Adding an Image Component**

1. Drag and drop an Image component from the Palette’s User Interface category to the Viewer.
2. Change the Image’s *Picture* property to the “HartfordMap.png” image that is provided in the template (or follow the directions below to create your own map image.)
3. Set the width to 100%
4. Set the height to 50%.

### Adding a ListPicker Component

The *List Picker* component can be used to select an option from a list of choices. The List Picker looks like a button but when clicked it displays a list of options for the user to choose from. The list can be a list of text or a list of numbers.

1. Drag and Drop a ListPicker component from the Palette’s User Interface category to the Viewer.
2. Set the List Picker’s *Text* property to say “Choose Destination”
3. Set the width to Fill Parent.
4. Set the height to Fill Parent.

To learn more about the List Picker, [read about the List Picker](http://appinventor.mit.edu/explore/content/basic.html#ListPicker) in the App Inventor glossary.

**The Activity Starter Component**

The *Activity Starter* component, which is built into the template, can be used to launch an application outside of the current app. This can be another application that is included on the device (such as the camera or Google Maps) or another App Inventor app that you’ve already installed on the device. (In the Android operating system an app is called an ***activity***. So in effect, the Activity Starter component is an app starter.) In order for Activity Starter to start an app, it needs to be told what type of activity to start and what type of action it will perform. We do this by setting several of its properties.

The Activity Starter component has already been incorporated into the template. It has been configured to interface with the Google Maps app on your Android device. The configuration involves the settings to the following three properties:

1. **Action.** The Activity Starter’s ***Action***property has been set to ***android.intent.action.VIEW***. This is specifying that we want to view (as opposed to edit or delete) something. .
2. **Activity Class.** The ***Activity Class***property specifies the name of the Java Class that implements the Google Maps app in the app’s source code. It is set to ***com.google.android.maps.MapsActivity***.
3. **Activity Package.** The package name is the name of the software package that implements the Google Maps app on the device. It has been set to ***com.google.android.apps.maps***.

Together these three properties tell the Android operating system what app should be started by the Activity Starter. To learn more about these properties, [read about the Activity Starter](http://ai2.appinventor.mit.edu/reference/components/connectivity.html) in the App Inventor glossary. There is one other property that we need to set when using the Activity Starter -- the ***DataUri.*** This will be set when we code the blocks to start up Google Maps.

# Coding the App’s Behavior

We will use the List Picker and the Activity Starter to open Google Maps and view a desired location on the map. To do this we will create a list of locations that the user can choose from.

## About Lists

The simplest data abstraction in programming is a variable, but there are more complex data structures available in all programming languages. App Inventor has a data structure called ***list*** that allows the storage of multiple items under one name in memory. The items are ***indexed*** which means they are ***numbered from 1 to the length of the list***. To define a list, we can create a global variable that can be initialized to an ***empty list*** (a list with no items on it):



Or we can assign the variable a specific list of items using ***make a list***:



The *Lists* drawer contains lots of blocks ([List blocks](http://appinventor.mit.edu/explore/ai2/support/blocks/lists.html)) such as *insert item into list* and *select random item from list* that let you manipulate the items in the list.

### AP Pseudocode

In the AP CSP pseudocode, lists are represented using square brackets [ ] as shown below. The assignment operator ← (the left-pointing arrow) can be used to assign a list to a variable. So the initialization of the global *destinations* variable in App Inventor would look like this in the AP pseudocode:

# destinations ← [ "Connecticut State Capitol", “Hartford Atheneum", “Trinity College”]

List items can be numbers or text or other lists. Text items are also called ***strings***, which are usually indicated by quotes "" to distinguish them from variables.

## Creating a List of Destinations

Begin by creating a *list* of places that you might want to visit while in your vicinity.

1. Initialize a list variable called *destinations* by using an *initialize global variable* block from the Variables drawer and a *make a list* block from the *Lists* drawer in the Toolbox
2. Use *text* blocks to add items to the list. If you need more than two items, remember you can use the mutator (the blue widget) to open up more items on the list. In our example we are using Hartford, Connecticut locations:



**Setting the List Picker’s Elements**

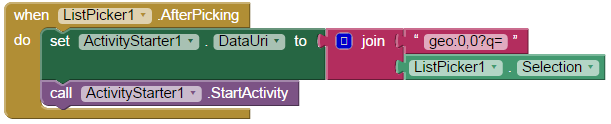
Now that you have a list of destinations, you’ll want to display this list to the user. Here is where you should make use of the List Picker. When the screen initializes, set the List Picker’s *elements* to be the global list variable *destinations*. This will make each item in *destinations* a choice in the List Picker. Your code should look like this:

****

**Opening Another Application with the Activity Starter**

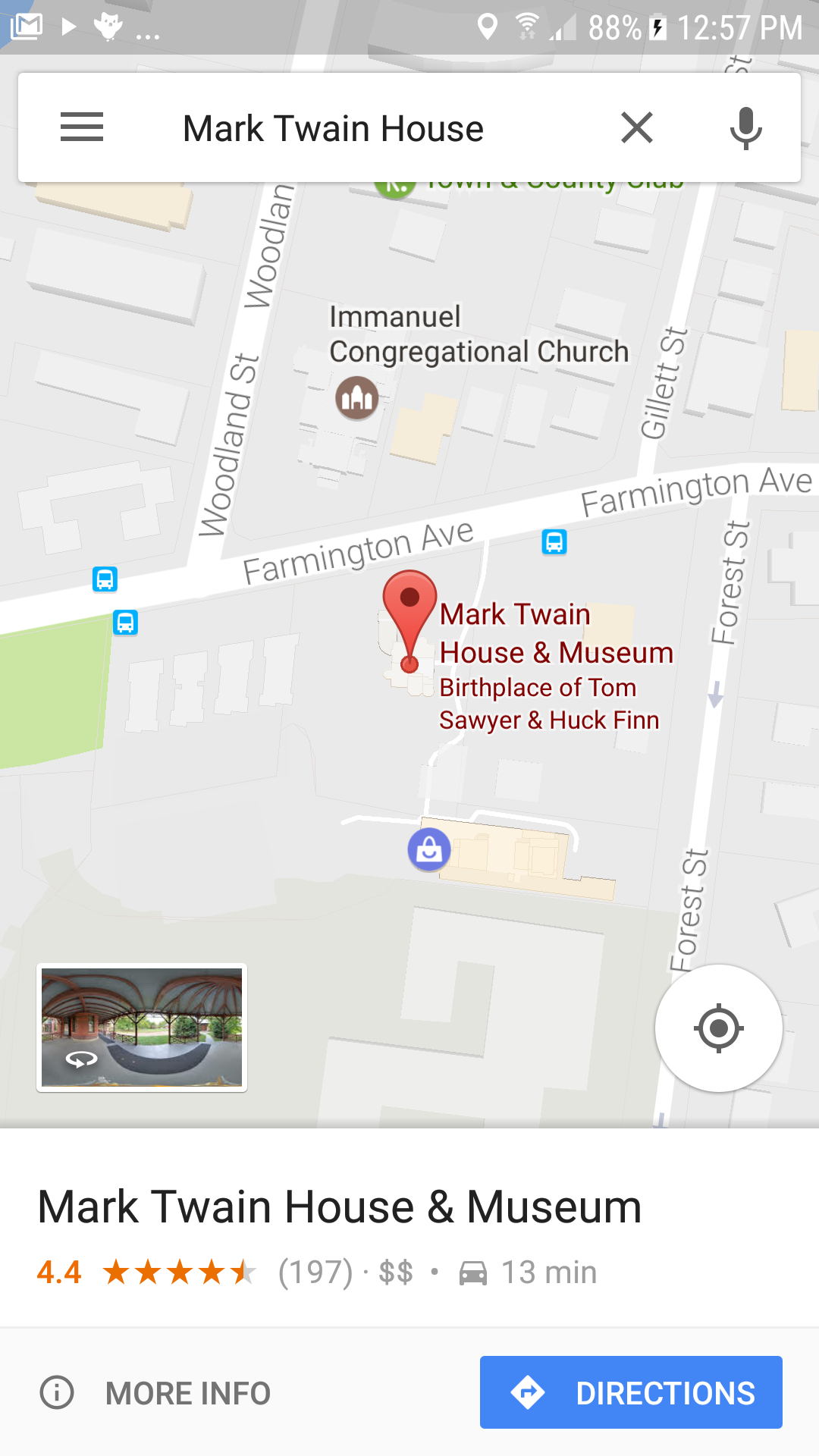
So far you have created a list of possible destinations and then displayed that list in the List Picker. When the user selects one of the destinations in the List Picker the app should open the selected location in Google Maps. To do so, we use the ***ListPicker.AfterPicking*** event handler.

When you’re done coding the *AfterPicking* handler, it should look like this:



Here are the steps.

1. Get a *ListPicker1.AfterPicking* event handler from the ListPicker1 drawer in the Toolbox
2. Get a setter block from the ActivityStarter1 drawer in the Toolbox and change the mutator to select the *DataUri* property. **The *DataUri* property specifies data used to tell Google Maps what to display.**
3. Get a *join* text block from the Text drawer in the Toolbox.
4. Get an empty string text block from the Text drawer and type “**geo:0,0?q=**” without the quotes. (You may copy and paste.) This is the code that is used for *url encoding.*
5. Get a getter block from the ListPicker1 drawer and change the mutator to select the *Selection* property.
6. Get a call *ActivityStarter1.StartActivity* block from the ActivityStarter1 drawer. This block launches the activity that is specified with the activity starter. In Map Tour, this will open the chosen destination in Google Maps. You can [read more here](http://ai2.appinventor.mit.edu/reference/components/connectivity.html) on how to use the Activity Starter to launch other activities.

For example, if the user selected “Mark Twain House”, this code will concatenate ***geo:0,0?q=Mark Twain House,*** which tells Google Maps to display a pin at the Mark Twain House on a Google map. The result will be the map shown in the image on the left.

## Run and Test the App

You’re ready to test your app on your device (phone or tablet) or on the emulator. As you select destinations from the List Picker, the app should display a map with a pin at the selected location.

**What’s Happening Here:** When the user selects a destination the app is starting the Google Maps app on the device. So now there (at least) two apps running on the device: (1) Map Tour and (2) Google Maps. The Google Maps app is in the foreground and the Map Tour is in the background. Hitting the device’s *Back Key* will bring you back to the Map Tour app.

If Google Maps can’t find a location, it will display a map and a *“No results found”* message. Check the spelling of your destination and make sure it’s a real destination in your vicinity.

## Picking a Random Destination

Because we have stored our destinations in a list, App Inventor has a block that makes it is easy to select a random destination from the list. Just use the ***pick random item*** block in the *List* drawer to select a random item from the destinations list:



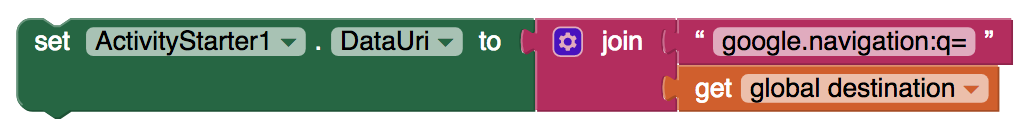
Let’s add a button to the UI that will bring the user to one of the random destinations on the list. We’ll name the button *ButtonSurpriseMe* and when it is clicked it should set the *DataURI* to a random destination from the list and start the Activity Starter:



## Asking Google Maps for Directions

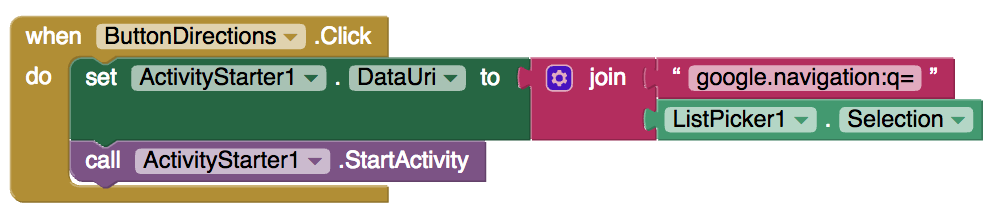
As we noted above, this app is making use of Google Maps ***Application Programming Interface (API).*** An [API](https://en.wikipedia.org/wiki/Application_programming_interface) is a set of software tools that programmers use in building applications, such as you are doing. Learning how to use an API lets you take advantage of software that other programmers have created to do some pretty cool things.

For example, the Google Maps API lets you ask for directions. In that case the ***DataUri*** that you have to provide looks like this, where the global *destination* variable is storing a location such as “Mark Twain House”.



By default, Google Maps will provide navigation instructions to the specified destination starting from your current location -- assuming your device has GPS.

Let’s add a *ButtonDirections* button to the app and code it to tell Google Maps to provide directions to a previously selected destination. Code the *ButtonDirections.Click* handler as shown here:



This will provide directions to the destination that was last selected with the List Picker.

# Enhancement Mini Projects

1. Add your own map image to the UI and add your own locations to the *destinations* list. **Directions:** Click here to open [Google Maps](https://www.google.com/maps) in your browser and search for your town or city. Take a screenshot of a portion of the map of your vicinity. On Mac you can use the *Preview* program to do this. On Windows machines you can [follow these instructions](https://www.howtogeek.com/226280/how-to-take-screenshots-in-windows-10/). Save the screenshot on your computer. Upload the screenshot using the *Upload File* button in the *Media* panel:
2. Try some of the other commands that come with the Google Maps API. Among other things, you can control the type of directions (by foot or car or bicycle), the type of map (street view, satellite view, hybrid) and many other things. Here’s a link to the [API documentation](https://developers.google.com/maps/documentation/urls/android-intents). And here are some example URIs to try:

|  |  |
| --- | --- |
| Find restaurants in the vicinity | geo:0,0?q=restaurants |
| FInd restaurants in Hartford | geo:41.7618,-72.6806?q=restaurants |
| Display street view of Hartford | google.streetview:cbll=41.7618,-72.6806 |

# Still Curious? APIs Extend Your Powers as a Programmer

In this app, you made use (through Activity Starter) of Google Maps, an existing mobile app on your device. You used the Google Maps [Application Programming Interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API) to control the maps that were displayed in your app. The [Google Maps API](https://developers.google.com/maps/) provides documentation for programmers and app developers on how to interact with its application. There are lots of APIs available to programmers. Their role is to specify exactly how programs and apps can interact with each other to perform certain tasks, like sending email or Twitter messages or displaying a map. The API specifies exactly what information you need to provide and in what specific format to provide it in order to interact with an existing application.

One interesting implication of this lesson is that APIs enable programmers to see the Internet and Web and their mobile devices in a very different way than other users. Rather than seeing it merely as something to use, APIs allow programmers to ***control*** how they interact with their mobile devices and with applications provided by Google, Amazon, Twitter, and others.

***Complete the Self-Check Exercises and Portfolio Reflection Questions as directed by your instructor.***