



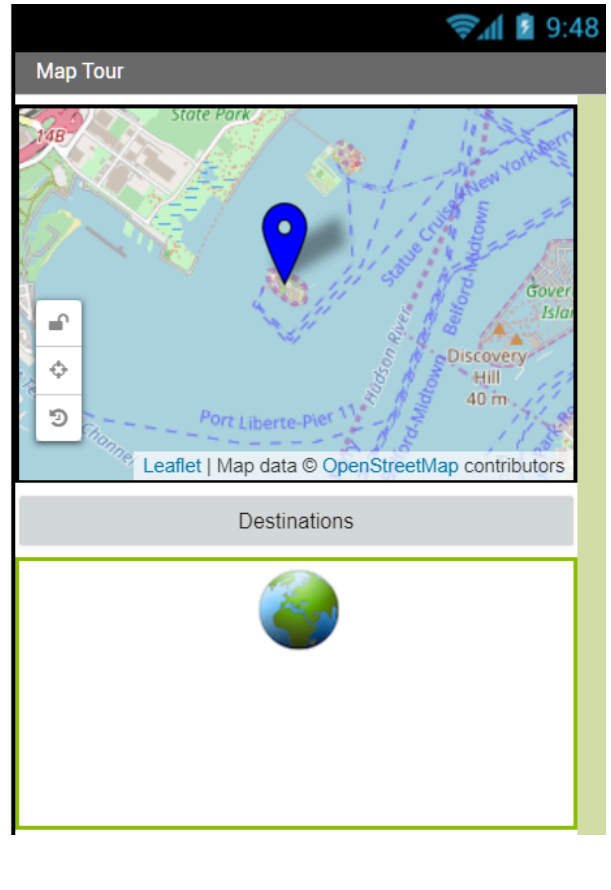
Overview: Map Tour Tutorial

The Map Tour allows a user to select a location from a list and show the selected location on the map using App Inventor's built-in Map component.

Objectives: In this lesson you will create an app that:

- uses the new built-in **Map** component in App Inventor.
- uses **Lists** and **ListPickers** to store and access a list of destinations on the map.
- uses an **Application Programming Interface (API)** to display a Wikipedia page in a Web Viewer

[Short Handout](#)



Getting Started

Open App Inventor and create a new project called Map Tour. Decide on 3 destinations/landmarks for your Map Tour. Choose well-known locations that appear in Wikipedia. These could be landmarks in your town or state or even in another country.

User Interface



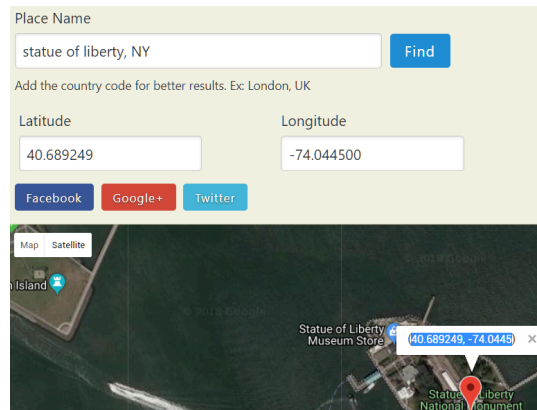
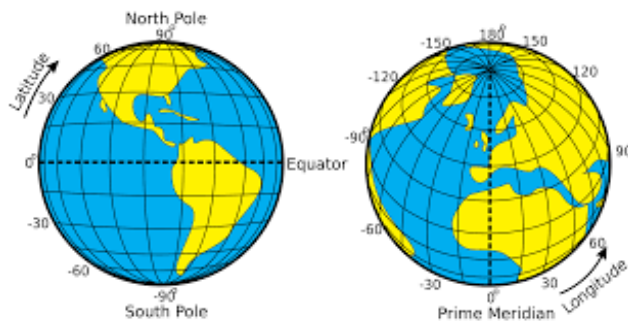
This app uses the new App Inventor [Map component](#) (which is built using OpenStreetMap) to display maps. It also uses a component called [List Picker](#) which looks like a button but lets the user pick an item from a List.



Map Component

Drag in a Map component from the Maps drawer and set its height to 50 percent and width to fill parent. In its properties, try the drop down menu for different Map Types and the checkboxes to Show Compass, Show User, Show Zoom.

The Map's CenterFromString property contains the latitude and longitude for a location to put at the center of the map. Latitudes and longitudes identify locations anywhere in the world. Go to <https://www.latlong.net/> to find the lat/long for your town (some alternatives to this website are <https://gps-coordinates.org/> and maps.google.com which will give you the latitude and longitude if you right click on any location and click on "What's here"). Copy the latitude, longitude string inside the parentheses from the location in the map at this website into the CenterFromString property in App Inventor. For example, the string for the Statue of Liberty in NY is `40.689249, -74.0445` as seen below.

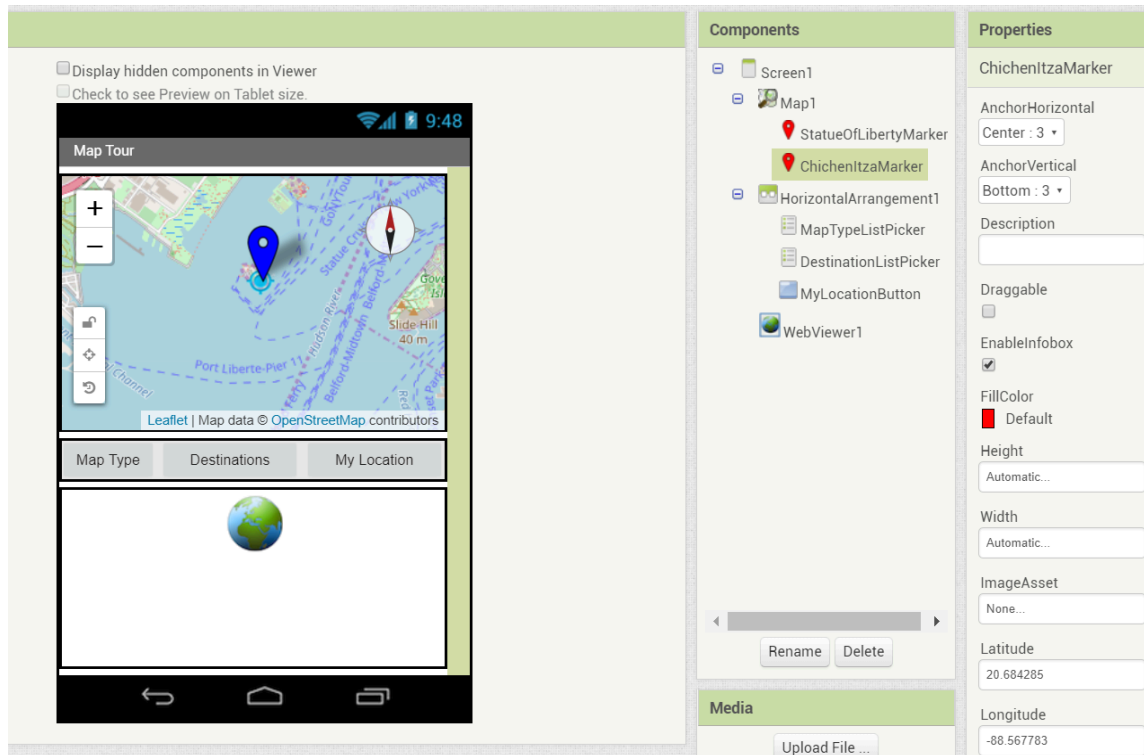


Markers

Choose three destinations or landmarks for your Map Tour. Try to choose destinations that are well-known so that they have a page in Wikipedia. Drag in two or three markers on to the map from the Maps component. These will be the destinations in the Map Tour.

- Rename each marker for different locations or landmarks.
- Check off Enable InfoBox and type in a title property for each marker.
- If you want to use destinations that are far apart and do not fit on the initial map, look up the latitude and longitude for your destination at <https://www.latlong.net/> and copy them into the latitude and longitude properties of a marker.

For example, here we have a StatueOfLibertyMarker and an off screen marker for the Chichen Itza Mayan ruins in Mexico where we copied in the latitude 20.684285 and longitude 88.567783.



ListPickers

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. This list can be set in the UI in the **ElementsFromString** property or in the code in a BeforePicking event.

Drag in a List Picker and rename it ListPickerDestinations with the text Destinations and a width of Fill Parent.

WebView:

Add a [Web Viewer](#) Component to the bottom of the UI and set its width to fill the parent. This will display the Wikipedia information about each chosen landmark.

UI Summary:

Here is a table summarizing all the UI Components for this project.

UI Component	Name	Properties
Screen	Screen1	<ul style="list-style-type: none"> Background Color of your choice
Maps/Map	Map1	<ul style="list-style-type: none"> Height - 50 percent



		<ul style="list-style-type: none"> • Width - fill parent • Check Show Compass, Show User, Show Zoom • Try different Map Types • Center from String: set to a latitude, longitude using https://www.latlong.net/ as explained below.
Maps/Marker	Marker1, Marker2, Marker3 (rename with name of your locations)	<ul style="list-style-type: none"> • Check Enable InfoBox • Title - Name of your location • Set latitude and longitude from https://www.latlong.net/ if you do not see its location initially on the map.
ListPicker	ListPicker1 - Rename DestinationListPicker	<ul style="list-style-type: none"> • Text - Destinations • Width - Fill Parent
User Interface / WebView	WebView1	<ul style="list-style-type: none"> • Width - Fill Parent

Coding the App

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 an **Abstract Data Type (ADT)** 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*:



Notice that a variable in App Inventor can hold a single data item like a number or a whole list containing many items. Actually, variables in App Inventor can hold a variety of **data types** including:

- Numbers: integers or decimal numbers,
- Strings: text, any sequence of characters you can type on a keyboard, represented



inside quotes like "Hello World! 123".

- Booleans: like true or false
- Lists: a collection of related items given a name. The items can be any data type but they are usually all the same data type, for example all strings or all numbers.

The *Lists* drawer contains lots of blocks ([List blocks](#)) such as *insert item into list* and *select item from list* that let you manipulate the items in the list.

AP Pseudocode

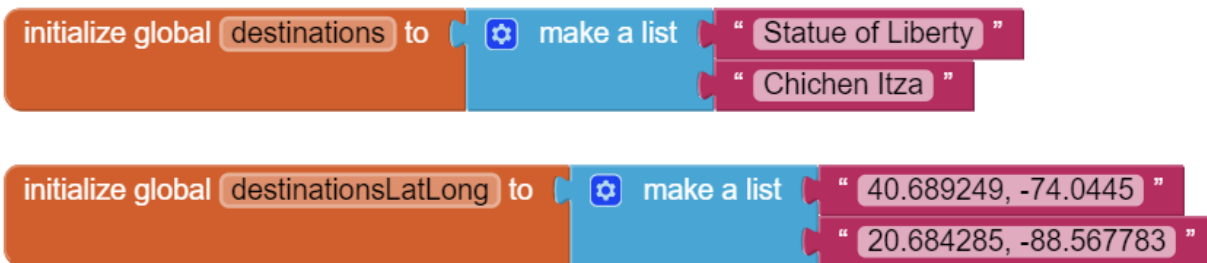
In the AP CSP pseudocode, lists are represented using square brackets [] as shown below. The assignment operator ← (the left-pointing arrow) can be used to assign a list to a variable. The initialization of the global *destinations* variable in App Inventor would look like this in the AP pseudocode:

`destinations ← ["Statue of Liberty", "Chichen Itza"]`

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.

Two Parallel Lists

The lists will hold the information about the destinations/landmarks that you made with the markers on your Map Tour. Because OpenStreetMap uses Latitudes and Longitudes instead of place names, we will need two parallel lists, one with location names like "Statue of Liberty" and the other with the latitude and longitude of each corresponding location. You will need to look up each location's latitude and longitude in <https://www.latlong.net/> or from your markers and copy them into your code. Here are two example lists where the first location name in the destinations list matches with the first latitude/longitude string in the destinationsLatLng list:



Here is a summary of the data variables that you need using the make a list block.

Abstraction: List Variables	Values
destinations	A list of destinations created using a make a list block, for example "Statue of Liberty", "Chichen Itza"



destinationsLatLng	A list of latitude, longitude strings corresponding to the destinations in the destinations list, for example ["40.689249, -74.0445", "20.684285,-88.567783"] which correspond to the latitude, longitude string for the Statue of Liberty and Chichen Itza.
--------------------	--

DestinationListPicker

To code the Map Tour, we need to set the DestinationListPicker's Elements of the LocationListPicker to the first list, destinations, so that it will display that list to the user

```
when DestinationListPicker .BeforePicking
do set DestinationListPicker . Elements to get global destinations
```

After the user selects one, you can get which element they chose with the ListPicker.Selection block or the index (number) of the element they chose using the ListPicker.SelectionIndex block.

```
DestinationListPicker . Selection
DestinationListPicker . SelectionIndex
```

Then, you can use the built-in list [select list item from list](#) block to find the corresponding latitude and longitude from the parallel destinationsLatLng list and set the Map's CenterFromString property to it.

```
select list item list
index
```

The resulting code is:

```
when DestinationListPicker .AfterPicking
do set Map1 . CenterFromString to select list item list get global destinationsLatLng
index DestinationListPicker . SelectionIndex
```

WebView

We can go to any website in the [WebView](#) component using the **GoToURL** block and a Text block with the URL (web address). For example, the wikipedia site:



```
call WebView1 .GoToUrl
    url "http://en.wikipedia.org/wiki/"
```

If you go to Wikipedia.org and search for something, for example Statue of Liberty, you'll see that it places you in the page https://en.wikipedia.org/wiki/Statue_of_Liberty. So, if we join together the ListPicker Selection with the wikipedia url, we can get to each destination's wikipedia page (if there is one).

```
when DestinationListPicker .AfterPicking
do
  set Map1 .CenterFromString to
    select list item list
    index
  get global destinationsLatLng
  DestinationListPicker . SelectionIndex
  call WebView1 .GoToUrl
    url join
      "http://en.wikipedia.org/wiki/"
      DestinationListPicker . Selection
```

This code works in most Android devices, however for some devices, such as iOS devices, you may need to replace all spaces in the Selection with underscores for the Web Viewer to find the page (so that "Statue of Liberty" becomes "Statue_of_Liberty"). For this, find the Text/Replace block and put in a Text empty string block and type in a single space in there for what to replace and an "_" for what to replace with. Here's the resulting code:

```
when DestinationListPicker .AfterPicking
do
  set Map1 .CenterFromString to
    select list item list
    index
  get global destinationsLatLng
  DestinationListPicker . SelectionIndex
  call WebView1 .GoToUrl
    url join
      "http://en.wikipedia.org/wiki/"
      replace all text
        segment
        replacement
      DestinationListPicker . Selection
      Type in a space
      Type in an underscore _
```

The **API (Application Programming Interface)** for Wikipedia allows programmers to get to specific pages by putting the page name in its URL. Many web services (Facebook, Twitter, Google Maps, Wikipedia, etc.) have APIs that allow access to their service for programmers. This simplifies complex programming tasks and allows communication between different programs.

A summary of your code is below:



Event Handlers	Algorithms
DestinationsListPicker.BeforePicking	Set DestinationsListPicker.Elements to the destinations list
DestinationsListPicker.AfterPicking	-Set Map1.CenterFromString to select list item from destinationsLatLng list using DestinationsListPicker.SelectionIndex (see below). -Call WebViewer1.GotoURL and join the text " https://en.wikipedia.org/wiki/ " and DestinationsListPicker.Selection

Testing the App

Test your app with the following inputs. See if the outputs match the expected outputs.

Inputs	Expected Outputs	Actual Outputs
Pick a destination	The map should change to the picked destination and the Web Viewer should change to the Wikipedia page for the destination if it exists. Make sure you try all on your list.	?

Enhancements

Your instructor may ask you to do some or all of the following enhancements for your Map Tour app. Be creative!


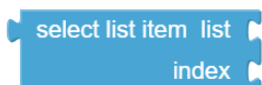
1. Add more destinations to your map tour. Make sure you have at least 3 destinations.
2. **MapType ListPicker:** Add a ListPicker to choose the Map Type with the Elements Roads, Aerial, and Terrain. These elements can be set in the UI or in the code in the BeforePicking event handler. After picking, use the user's **Selection** or **SelectionIndex** to set the Map.MapType to 1 for Roads, 2 for Aerial, and 3 for Terrain. You could do this with an if block using the blue mutator button to add if/elseif/else parts to make a 3 way choice.
3. **Zoom Slider:** Add a slider to your UI to control the zoom level in the map. You may want a horizontal arrangement to arrange these new controls. In the slider's properties, set the MaxValue to 20, MinValue to 1, and ThumbPosition to 13. The slider has a When Slider Position Changed event handler that is called when the user slides the slider. Inside this event, you can change the Map1's Zoom property to value in the Slider's ThumbPosition.
4. **My Location button and GPS:** OpenStreetMap keeps track of the user's location using GPS (The Global Positioning System which allows people to pinpoint their geolocation (geographic location) on Earth using satellites). The Map's properties UserLatitude and UserLongitude will give the latitude and longitude of the device currently running your app if the device has GPS capabilities. Add a button called My Location. When it is clicked, use the Map.PanTo procedure to go the the Map's UserLatitude, UserLongitude, Map.ZoomLevel. **NOTE:** This enhancement is very dependent on the type of device you have and where you are -- being indoors in a classroom is not optimal. So to get this part of the app working you may want to package the app and take the device outdoors.



Also, make sure that the device's GPS sensor is on.

Summary

Here are some new App Inventor List blocks that you used in this lesson and their AP pseudocode versions for the AP exam.

App Inventor Blocks	AP Pseudocode
	<code>destinations ← ["Statue of Liberty", "Eiffel Tower"]</code>
	<code>list[index]</code>

Vocabulary Review

Review the following new vocabulary in this lesson. If you don't know what these mean, look back through the lesson.

- Data Abstraction
- Abstract Data Type (ADT)
- Data type
- List
- Index
- String, string concatenation (join)
- Boolean
- ListPicker
- WebView
- API
- GPS

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