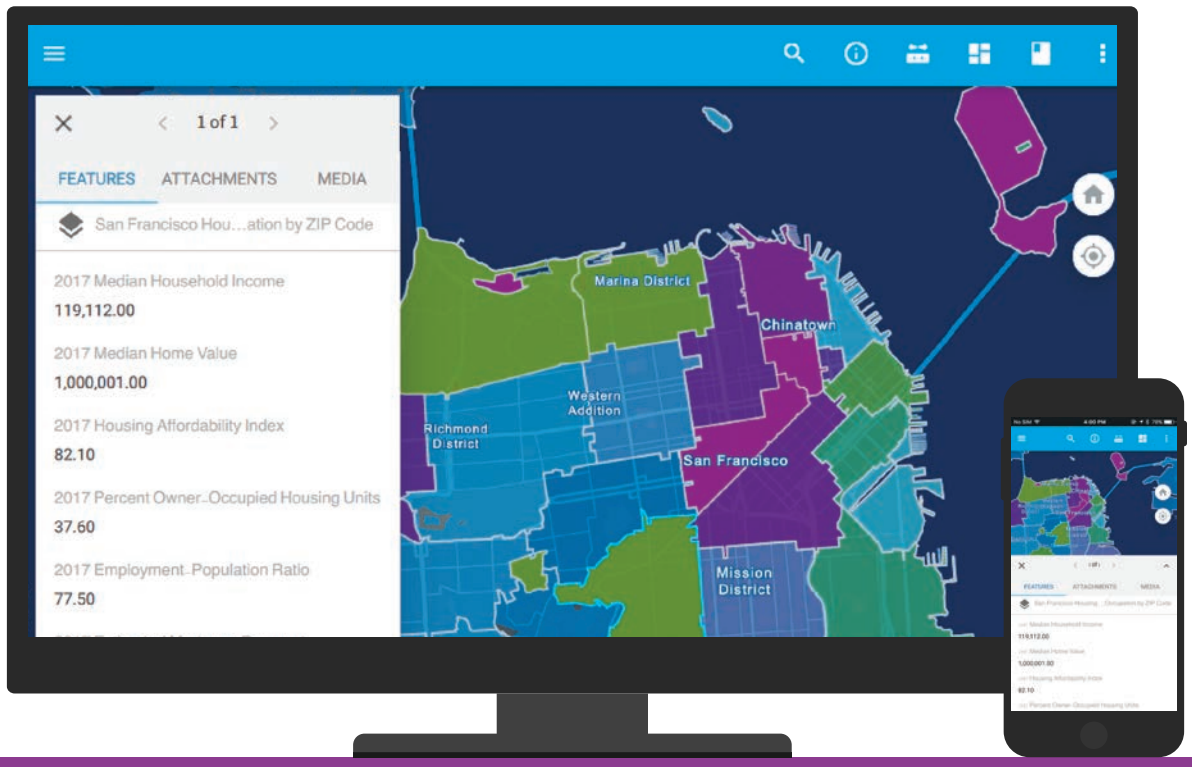
New AppStudio User Guide to

Building a **Hello, World** App

AppStudio
for ArcGIS



esri THE SCIENCE OF WHERE™

Hello, World: A Tutorial

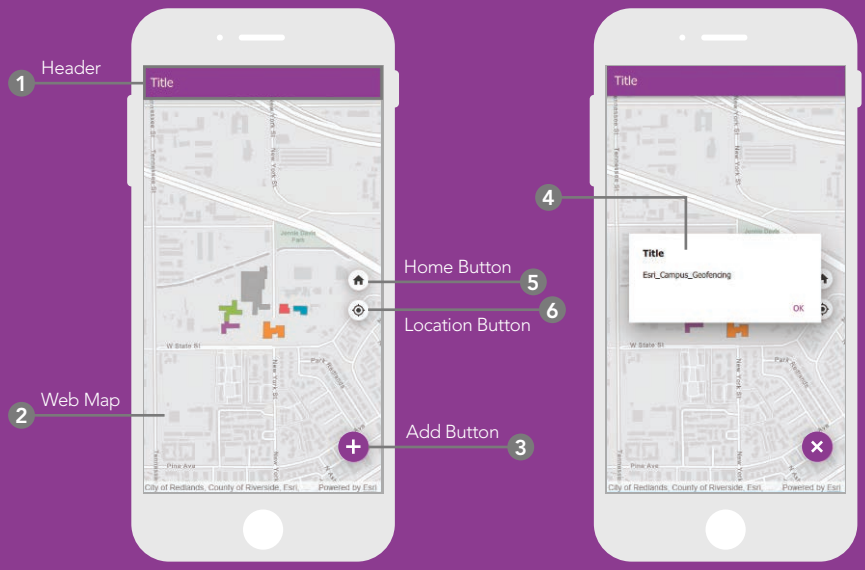
In this tutorial, you will learn how to use AppStudio for ArcGIS to quickly create a native app that loads a public web map and displays a device's current location on a map. You will use ArcGIS Runtime SDK for Qt for the mapping, one of the five ready-to-use material design layouts available for the app, and the device's GPS for location.

Time required: 10–20 minutes (if the prerequisite software is installed)

You will create an app that looks like the illustration below.
The main features of this app allow you to

- Load a public web map.
- Tap the Add button to display a web map name in a pop-up.
- Tap the Home button to return to the initial web map extent.
- Tap the Location button to display the device's current location.

Final result



Code behind

```
App{
    id: app
    width: 421
    height: 750
    // App Page
    Page{
        anchors.fill: parent
        // Adding App Page Header Section
        header: ToolBar{ id: header... } 1
        // TODO:-Add Code here
        MapView{
            Map{ id: map... } 2
            ColumnLayout{
                FloatActionButton{ id: homeButton... } 5
                FloatActionButton{ id: locationButton... } 6
            }
            // set the location display's position source
            locationDisplay { ... }
        }
        // Button to perform action
        RoundButton{ id: addButton... } 3
    }
    // Show Dialog on the Add button click
    Dialog { id: messageDialog... } 4
}
```

Getting ready

- You will need the AppStudio for ArcGIS Standard license. If you are new to AppStudio or have not yet gotten an AppStudio Standard license, please watch this tutorial to get started with a trial account.
- You will need to download resources (see attachment) to follow the tutorial.

Step 1. Create a new app

You will start from one of the readily available layouts in AppStudio, called Simple Layout (Material Design):

a. Start AppStudio.

b. Click the **New App** button.

c. Click the **Layout** tab.

d. Choose Simple Layout (Material Design) and click the **Create** button.

Once the app is created, click the **Edit Title** button at the top of the side panel to rename the app. Name it My First App. Click the **Save Changes** button.

e. Click the **Edit** button on the side panel to open the Qt Creator integrated development environment (IDE).

f. By default, the first QML file used by the app will be loaded into the Editor (MyApp.qml).

Step 2. Add import statements

You can tell the app which modules, components, or plug-ins you want to use by adding an import statement. For example, to use the material style in the app, you need to add the import statement **import QtQuick.Controls.Material 2.1**. In this step, you will add three more import statements to use ArcGIS Runtime as well as enable location-based functionality.

```
//ArcGIS Runtime
import Esri.ArcGISRuntime 100.1

//GPS
import QtPositioning 5.8
import QtSensors 5.2
```

Step 3. Change primary color property

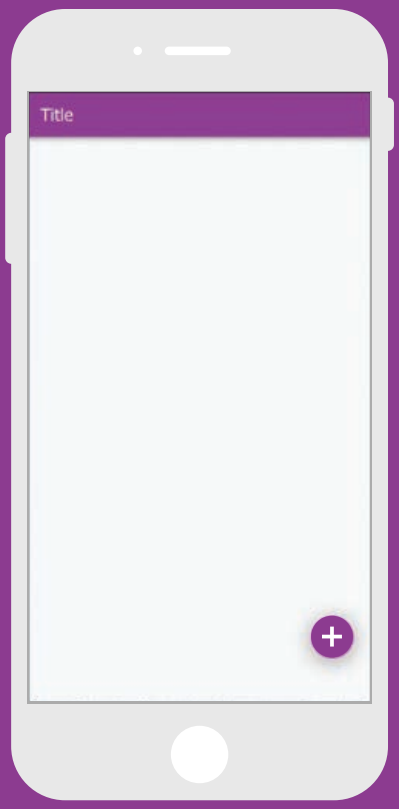
Material primary color can be used to develop your branding color. Material accent color holds the accent of the theme. It can be the same as the primary color. In this step, you will change the primary color from "#3F51B5" to "#8f499c".

```
readonly property color primaryColor: "#8f499c"
```

Note

Below are the keyboard shortcuts to save changes:
Windows: Ctrl+S
Mac OS: Command+S

Below are the keyboard shortcuts to run apps in Qt Creator:
Windows: Shift+Alt+R
Mac OS: Shift+Option+R



Step 3. Your app should look like this.

Step 4. Add a header

The simple material design layout already has a header; therefore, you do not need to add any more code in this step. Note that ToolBar QML type is used to define the header. The header height is 56 units, and the background color is Material.Primarycolor. Inside the ToolBar, RowLayout QML type was used to position the contents.

Step 5. Load a public web map

In this step, you will add a map to the app by using ArcGIS Runtime SDK for Qt. Remember that the import statement was added in step 2.

MapView is a virtual class from ArcGIS Runtime SDK that renders data in a map and allows users to interact with the map. The initUrl property inside the Map QML type is used to initialize the web map. The initUrl can be the item details page URL, the map view page URL, or the sharing data URL.

Add the following code below the comment // TODO:-Add Code here to the MyApp.qml file:

```
MapView{
    id:mapView
    anchors.fill: parent
    Map{
        id:map
        initUrl: "http://www.arcgis.com/home/item.html?id=7811ccdb71c84635a7a42eb020403615"
    }
}
```

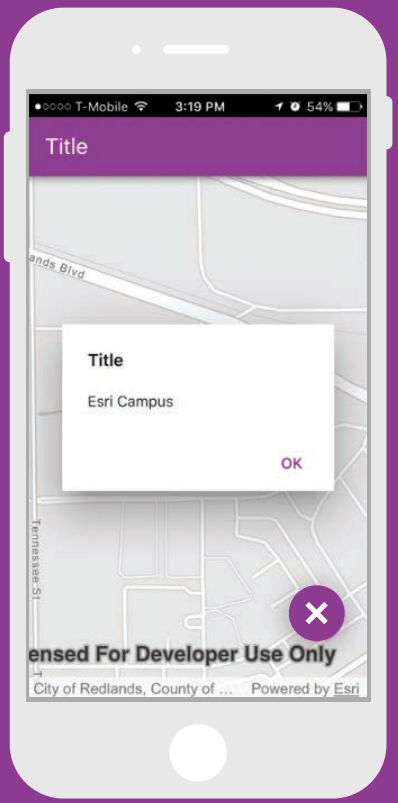
Note

The style of double quotes copied from this PDF file might not be supported in Qt Creator IDE. Please type double quotes for initUrl in the IDE.

Step 6. Add the web map name to the pop-up

You will replace the default text shown by clicking the **Add** button with the web map title. To do this, go to the **Dialog** component and replace the Label text property with the code below. Note that the value is a property binding to the item title of the web map as loaded by the web map. (This is really awesome and so convenient. QML rocks!)

```
text: qsTr(mapView.map.item.title)
```



Step 6. Your app should look like this after you click the ✕ button.

Step 7. Add the FloatActionButton component

Now you'll change gears a bit and structure the app by using reusable components for the user interface (UI), like buttons.

In QML, any snippet of code may become a reusable component. A file-based component is created by putting a QML element into a separate QML file. The file name is the element name, and you can use the component by calling its file name. A file-based, reusable FloatActionButton component has been created for the Home and Location buttons. The attached folder contains the reusable component file FloatActionButton.qml. Once you have downloaded the attachment, you can add this file to the app project. To do this, go to AppStudio Desktop and choose the app that you are working on:

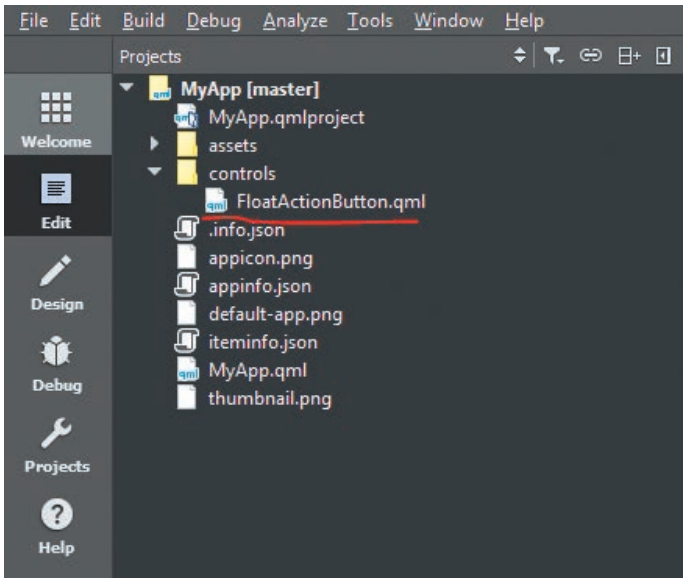
a. Click the **Files** button on the side panel.

b. Create a new folder and name it controls.

c. Copy and paste the FloatActionButton.qml file to the controls folder.

Now that the FloatActionButton component has been added to the project, you might be wondering how to use it. First, you need to add the import statement. After that, you can directly call the file name—which has the same name as the component FloatActionButton in the main file (MyApp.qml)—to use the component.

```
import "controls"
```



Step 7. You should now be able to see this in Qt Creator.

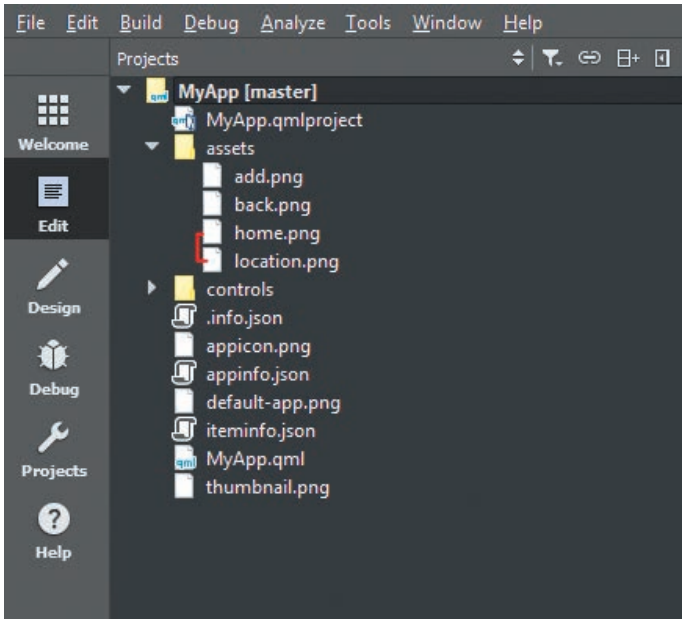
Step 8. Add the Home button and Location button images

The next step is adding the Home button and Location button images to this project (you will find the images in the attached folder):

a. In AppStudio, choose the app that you are working on.

b. Click the **Files** button on the side panel.

c. Copy and paste home.png and location.png to the assets folder.



Step 8. You should now see this in Qt Creator.

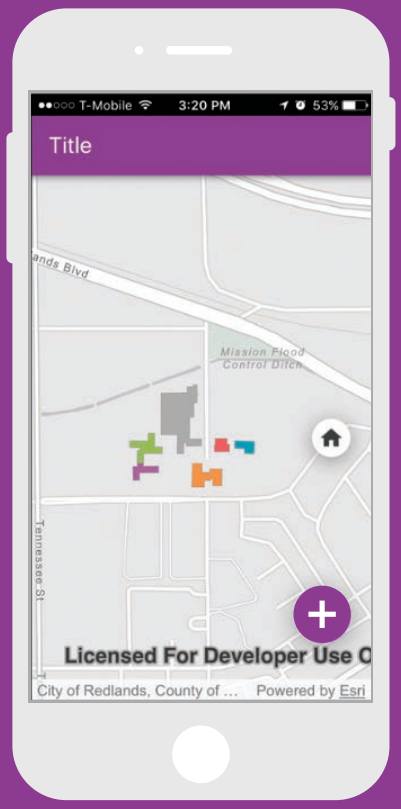
Step 9. Add the Home button

You will use ColumnLayout to position the Home and Location buttons by stacking them one on top of the other. Add the following syntax inside the MapView QML type and below the Map QML type.

```
ColumnLayout{
    anchors{
        right: parent.right
        rightMargin: 16 * scaleFactor
        verticalCenter: parent.verticalCenter
    }
}
```

Inside the ColumnLayout QML type, add the Home button by calling the name of the **FloatActionButton** component. Add the code below. Note that when clicking the Home button, you will call the setViewpointWithAnimationCurve method to return to the initial web map extent. The first parameter of this method, **map.initialViewpoint**, sets the viewpoint. The second parameter, **1.0**, is the number of animation seconds, and the third parameter, **Enums.AnimationCurveEaseInOutCubic**, is a type of animation.

```
FloatActionButton{
    id:homeButton
    imageSource: "./assets/home.png"
    onIconClicked: {
        mapView.setViewpointWithAnimationCurve(map.initialViewpoint, 1.0, Enums.AnimationCurveEaseInOutCubic)
    }
}
```



Step 9. Your app should now look like this.

Step 10. Set up the location source

In this step, you will set up a location source by adding LocationDisplay QML Type. PositionSource is used to provide the device's current position. The import statement for this was already added in step 2. Insert the following code below the ColumnLayout QML type:

```
locationDisplay {
    positionSource: PositionSource {
    }
}
```

Step 11. Add the Location Button

The last step is to add a Location button beneath the Home button (homeButton) by adding the code below. Note that there are different location display modes in ArcGIS Runtime. In this case, you will use **Enums.LocationDisplayAutoPanModeRecenter**.

```
FloatActionButton{
    id:locationButton
    imageSource: "./assets/location.png"
       onIconClicked: {
          if (!mapView.locationDisplay.started) {
              mapView.locationDisplay.start()
              mapView.locationDisplay.autoPanMode = Enums.LocationDisplayAutoPanModeRecenter
              colorOverlay.color = "steelblue"
          }else {
              mapView.locationDisplay.stop()
              colorOverlay.color = "#4c4c4c"
           }
    }
}
```

```
// TODO:-Add Code here
MapView{
    id:mapView
    anchors.fill: parent
    Map{
        id:map
        initUrl: "http://www.arcgis.com/home/item.html?id=7811ccdb71c84635a7a42eb020403615"
    }
    ColumnLayout{
        anchors{
            right: parent.right
            rightMargin: 16 * scaleFactor
            verticalCenter: parent.verticalCenter
        }
        FloatActionButton{
            id:homeButton
            imageSource: "./assets/home.png"
            onIconClicked: {
                mapView.setViewpointWithAnimationCurve(map.initialViewpoint, 1.0, Enums.Anima
            }
        }
        FloatActionButton{
            id:locationButton
            imageSource: "./assets/location.png"
            onIconClicked: {
                if (!mapView.locationDisplay.started) {
                    mapView.locationDisplay.start()
                    mapView.locationDisplay.autoPanMode = Enums.LocationDisplayAutoPanModeRec
                    colorOverlay.color = "steelblue"
                }else {
                    mapView.locationDisplay.stop()
                    colorOverlay.color = "#4c4c4c"
                }
            }
        }
    }
    locationDisplay {
        positionSource: PositionSource {
        }
    }
}
```

Step 11. Your MapView component should look like this.

Since this app is based on ArcGIS Runtime, do not forget to add a free Runtime Lite license. (Check out the tip below—it's easy!)

Note

Set a free Runtime Lite license (remove watermark from the map):

1. Click the **Settings** button on the side panel.
2. Click the **Advanced** button.
3. Click the **Set Lite License** button and **Close Advanced Settings**.
4. Click the **Apply** button.

And you're done. It's that easy to create your own mapping application with AppStudio. Now you can use AppStudio Player to test the app on different devices. You can also use the Cloud Make tool to build executables of your app to distribute them in the app store or within your own enterprise/business.

esri® | THE SCIENCE OF WHERE™

appstudio.arcgis.com