QML and JavaScript for Native App Development

Michael Tims
Lucas Danzinger
Agenda

• Native apps. Why?
• Overview of Qt and QML
• How to use JavaScript skills to build native apps
Web vs Native vs Hybrid
Why build native apps?
Why build native apps?

- Work online & offline
- Access hardware and sensors
  - GPS, camera, Bluetooth, NFC, etc
- Great performance
- Publish apps to stores
The challenge

• Lots of platforms
• Lots of devices
• Unique development patterns
  - Languages
  - Frameworks
  - IDEs
  - Workflows

Platforms:
- iOS
- Android
- Windows
- Linux
- MacOS

Devices:
- iPhone
- iPad
- MacBook
- Galaxy S 5
- Nexus 10
- Surface

Languages:
- Swift
- Objective C
- Java
- Android
- .NET
- C#
- Cocoa

IDEs:
- XCode
- Android Studio
- Eclipse
- Visual Studio
- Gradle
You can build *native* apps with your *JavaScript* skills… and Qt!
What is Qt and QML?
And how is it related to JavaScript?
Qt and QML

- Cross platform framework
- Build native apps
- Same source code compiled for each platform
- Powered by C++ on the backend (very fast)
- Exposed through QML (based on JavaScript)
- ArcGIS Runtime SDK for Qt
  - Mapping API provided by Esri
  - Brings the power of ArcGIS to your devices
QML

Highly readable JSON/CSS-like syntax

Declarative UI elements

Imperative JavaScript Code to handle events

Dynamic property binding

ArcGIS Runtime

```qml
Rectangle {
  anchors.fill: parent

  Map {
    id: map
    anchors.fill: parent

    ArcGISTiledMapServiceLayer {
      url: "http://server.arcgisonline.com/ArcGIS/rest/services/World_Street_Map/MapServer"
    }
  }

  Button {
    anchors {
      left: parent.left
      top: parent.top
    }
    text: "Zoom to U.S."
    enabled: map.status === Enums.MapStatusReady
    onClicked: {
      var extent = {
        "xmin": -16780134, "ymin": -195554,
        "xmax": -4566023, "ymax": 8472000,
        "spatialReference": {"wkid": 3857}};
      map.zoomTo(extent);
    }
  }
```
JavaScript & QML

• If you understand JSON and JavaScript, this will come naturally
• Call JavaScript functions from QML
• Bind properties to JavaScript expressions
• Implements 5th edition of ECMA-262
• Access to the language standard types and functions
  - Object, Array, Math, and Date.
• JS is integrated in Qt Creator IDE
  - Intellisense in IDE
  - qmllint syntax tool
• Major Differences from the browser?
  - No DOM or window object
  - No Dojo, jQuery, or other frameworks
JavaScript Expressions
JavaScript expressions

- Bind properties to JS expressions
- Update UI automatically
- Any JS expression (no matter how complex) may be used in a property binding definition
- ex: Change UI depending on ambient light

```javascript
ArcGISTiledMapServiceLayer {
  url: darkGrayBasemapUrl
  visible: !isAmbientLightBright
}

ArcGISTiledMapServiceLayer {
  url: lightGrayBasemapUrl
  visible: isAmbientLightBright
}

GraphicsLayer {
  id: graphicsLayer
  renderer: isAmbientLightBright ? lightRenderer : darkRenderer
}
Updating the UI Automatically with Ambient Light Sensor
JavaScript Functions
JavaScript functions

- QML is declarative, but... You can (and will need to) call JavaScript functions from QML
  - Declare a visual component in QML (e.g. Button)
  - Write imperative JavaScript code to respond to an event
- “var” basic type used in functions (same as JavaScript var)
  - Can store numbers, strings, objects, arrays and functions
- Syntax based off ECMAScript 5th edition
  - Object, Array, Math, and Date
  - Syntax for if statements, switch statements, ternary operators, for loops, while loops, and more are no different than using JavaScript in the browser
function logInfo(callback) {
  var theArray = [];
  var theDate = new Date();
  for (var i = 0; i < 10; i++) {
    if (i !== 5)
      theArray.push("Item " + i);
  }
  console.log("There are", theArray.length, "items in the array");
  console.log("The time is", theDate.toString());
  console.log("The square root of 9 is:", Math.sqrt(9));
  callback();
}
Component.onCompleted: {
  logInfo(function() { console.log("callback fired") });
}
Asynchronous Programming with QML
Async programming with QML

- Signal and Handler Event System
- The event is a *signal* (ex: mouseClicked)
- The signal is responded to through a *signal handler*
  - Signal handler is the name of the signal with the “on” prefix (ex: onMouseClicked)
  - Write JavaScript to perform some procedure when a signal is emitted

```javascript
Map {
  // handle the mouseClicked signal on map with onMouseClicked handler
  onMouseClicked: {
    console.log("you clicked at", mouse.mapX, mouse.mapY);
  }
}
```
Using async JS functions for offline geocoding
Leveraging external JavaScript resources
Importing standalone JavaScript

- Separate non-trivial logic into external JavaScript files
- Make your code reusable
- How?
  - Place your functions inside a .js file
  - Import to QML files with import statement:
    ```javascript
    import "../Resources/DataDownload.js" as JS_Download
    ```
  - Execute your external functions from QML
    ```javascript
    JS_Download.downloadData(dataNeeded);
    ```
  - Include external JavaScript files from other JavaScript files with Qt.include()
Downloading items from ArcGIS Online for offline use
QML Canvas

• 2D and a 3D canvas item
  - Enables drawing via JavaScript
• Allows drawing of:
  - straight and curved lines
  - simple and complex shapes
  - graphs
  - referenced graphic images
• Convert HTML5 Canvas to QML
  - ex: charts.js
  - See Canvas QML Type doc

```javascript
ctx.beginPath();
ctx.moveTo(75,40);
ctx.bezierCurveTo(75,37,70,25,50,25);
ctx.bezierCurveTo(20,25,20,62.5,20,62.5);
ctx.bezierCurveTo(20,80,40,102,75,120);
ctx.bezierCurveTo(110,102,130,80,130,62.5);
ctx.bezierCurveTo(130,62.5,130,25,100,25);
ctx.bezierCurveTo(85,25,75,37,75,40);
ctx.closePath();
```
charts.js in QML using QML Canvas
How is Esri using QML?
How to get started?

- Download the sample viewer
- Install Qt and ArcGIS Runtime, and start building apps!
- Talk to us on GeoNet, Slack, Twitter, etc
- Come visit us at the showcase!
Recap

• Native apps. Why?
• Overview of Qt and QML
• How to use JS skills to build native apps
  - Property binding with JavaScript expressions
  - JavaScript functions
  - External JavaScript resources
  - Canvas
Related sessions

• Developing Cross-Platform Native Apps with AppStudio for ArcGIS (Advanced)
  - Wednesday 4:00 – 5:00 pm in Primrose A
• Cross-platform Native App Development with Qt/QML
  - Thursday 2:30 – 3:30 pm in Demo Theater 3
• Extending the Survey123 for ArcGIS Mobile App
  - Thursday 5:30 – 6:30 pm in San Jacinto Renaissance Hotel
Questions?