Collecting and Mapping Features with a GPS Watch and ArcGIS API for JavaScript 4

Prerequisite Software

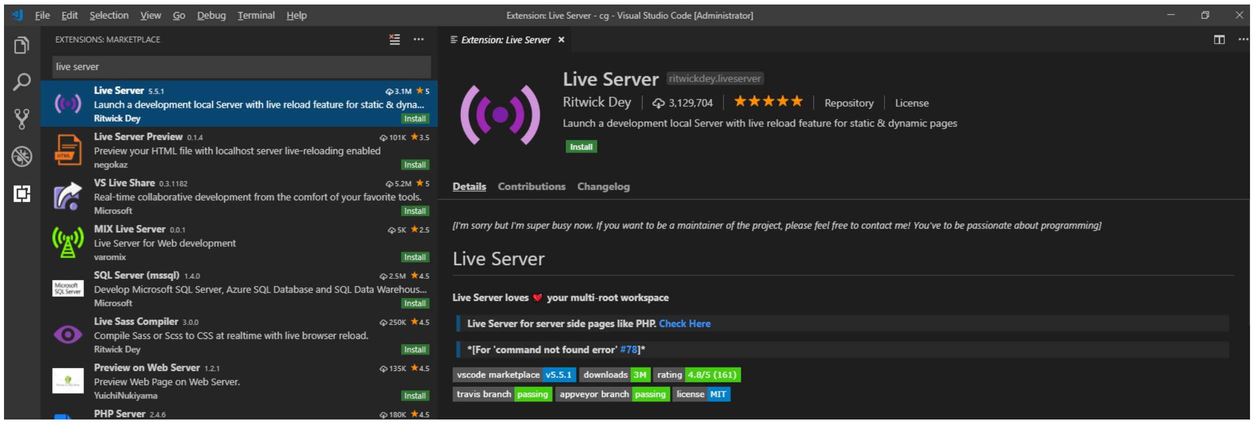
Visual Studio Code Setup

Installation

1. Go to: https://code.visualstudio.com/
2. Click the green Download button.
3. Select the Windows installer.
4. Run through the defaults.

Add Live Server Extension

1. Open the VS Code.
2. On the left navigation bar, click the Extensions (box in a box) menu item.
3. Search for "live server."
4. Click the green Install button.


File Edit Selection View Go Debug Terminal Help
Extension: Live Server - cg - Visual Studio Code [Administrator]
EXTENSIONS: MARKETPLACE
Extension: Live Server
Install
Repository
License
Details Contributions Changelog

live server

Live Server 5.5.1 — 3.1M ★5
Launch a development local Server with live reload feature for static & dyna...
Ritwick Dey

Live Server Preview 0.1.4 — 101K ★3.5
Preview your HTML file with localhost server live-reloading enabled
negokaz

VS Live Share 0.3.1182 — 5.2M ★5
Real-time collaborative development from the comfort of your favorite tools.
Microsoft

MIX Live Server 0.0.1 — 5K ★2.5
Live Server for Web development
varomix

SQL Server (mssql) 1.4.0 — 2.5M ★4.5
Develop Microsoft SQL Server, Azure SQL Database and SQL Data Warehous...
Microsoft

Live Sass Compiler 3.0.0 — 250K ★4.5
Compile Sass or Scss to CSS at realtime with live browser reload.
Ritwick Dey

Preview on Web Server 1.2.1 — 135K ★4.5
Preview Web Page on Web Server.
YuichiNukiyama

PHP Server 2.4.6 — 180K ★4.5

Live Server ritwickdey.liveserver
Ritwick Dey | 3,129,704 | ★★★★★ | Repository | License
Launch a development local Server with live reload feature for static & dynamic pages

[I'm sorry but I'm super busy now. If you want to be a maintainer of the project, please feel free to contact me! You've to be passionate about programming]

Live Server

Live Server loves ❤ your multi-root workspace

Live Server for server side pages like PHP. Check Here

[For 'command not found error' #78]

vscode marketplace v5.5.1 | downloads 3M | rating 4.8/5 (161)
travis branch passing | appveyor branch passing | license MIT

ArcGIS Online Account Setup

Create an ArcGIS Online Developer account.

Collecting Data using a GPX Watch

Note: If you prefer to not collect data and use existing point data, skip this section. TCX files are available in the Github repo.

Get a GPS Watch (Slide: Collecting Location Data using the GPS Watch)

1. Must be able to transfer the activity data off the watch. Mid-level GPS watches and above typically have Bluetooth and sync capabilities. (Garmin, Apple)

Use the GPS Watch to Collect Data (Slide: What Features to Collect?)

2. Decide which type of features point you want to collect. (Streetlights, mailboxes, trees, water fountains, houses, parks, scenic overlooks, a points of interest, etc.)
3. Start an activity. During the activity, press the lap button when you encounter a feature.
4. Complete one or more activities.

Download the Activity Data Files (Slide: Download from Watch/ Upload to Cloud)

Note: These instructions follow the Garmin ecosystem.

1. Follow the instructions to create a Garmin Account. https://connect.garmin.com/start/
2. Link your GPS watch to your phone using the Garmin Connect app.
3. Upload your activities from your watch into Garmin's cloud.
4. Login into your Garmin Account
5. On the left menu, select Activities…All Activities
6. Select the proper activity in the list.
7. On the activity page, click the gear icon in the top right, select Export to TCX. (The TCX file contains lap information. The GPX file does not.)
8. Repeat and download all activities you wish to include.

Transforming the Activity Files into GeoJSON

Get the Code

1. Go to the Url: https://github.com/jimlawruk/geodev2021
2. Clone or download the Repository into a geodev2021 folder on your desktop.

Examine the Activity Files

1. Open VSCode.
2. Open a TCX activity file
3. Take note of the timestamps, latitudes, longitudes, altitude, etc.
4. Search for <Lap . Count the number found to see many laps/points you have.

Examine the Code

1. In VSCode, select "File..Open"… select the folder called **TCXFileLapExtractor**. Click "Select Folder".
2. Open Program.cs. Note the instantiation of PointCollector and the process **ProcessActivityFiles()** is called.
3. Navigate to the **ProcessActivityFiles() method**. Note three methods to extract lap data, remove duplicates, and create GeoJSON files.

4. In PointCollector.cs, note the PointsCollected list, DuplicateThresholdDifference, and RemoveDuplicatePoints.

Process the Activity Files

5. Move or copy the TCX files into the **TCXFileLapExtractor** folder within the geodev2021 folder.

   The TCXFileExtractor is C# based program which processes each TCX activity file and collects all the coordinates from the start of each lap.

6. Open VSCode.
7. In the top menu, select Terminal, New Terminal.
8. In the terminal window, type: `dotnet run`
9. After a few seconds, you should see a new **lines.geojson** and **points.geojson** file in the directory.

Adjust Duplicate Processing

1. In ProcessActivityFiles(), set RemoveDuplicatePoints = false, then run the program. Notice how many more points are collected.
2. Revert RemoveDuplicatePoints = true. Adjust the DuplicateThresholdDifference value up and down. Notice differences in the number of points collected.

Publishing GeoJSON as a Feature Service in ArcGIS

1. Log into https://www.arcgis.com/.
2. In the very top navigation bar, select **Content**.
3. Click the **Add Item** button, select From Your computer.
4. Browse on your computer and select the *points.geojson* file.
5. Edit the title to *Camp Hill Street Lights*. (or something of your choosing)
6. Add a tag called street lights.
7. Click **Add Item**.
8. On the right, click the **Share** button. Select **Everyone (public).** Click **Save.**
9. In the bottom right, copy the Url of the new Feature Service, and paste in in a new browser window or somewhere to save it.
10. Repeat these steps for the *lines.json* file with the title like *Camp Hill Street Light Collection Routes*. (or something of your choosing)

Displaying Point Data on a Custom ESRI JavaScript API Map

Get the Code (If not done already)

1. Go to the Url: https://github.com/jimlawruk/geodev2021
2. Clone or download the Repository into a geodev2021 folder on your desktop.

Visual Studio Code

1. Install VSCode, with Live Server extension (see prerequisites above)
2. Open Visual Studio Code.

ArcGIS API For JavaScript

1. Go here to access the API Reference: https://developers.arcgis.com/javascript/latest/api-reference/
2. Use this a guide and reference for the properties, methods, samples, etc. of the various Modules, Widgets, Classes, etc.

Blank Map

Create a simple map using the ESRI ArcGIS JS API.

3. Copy the 0-blankMap.html to a new file called demo.html.
4. In the top menu, select File, Open Folder. Select the geodev2019 folder.
5. In the explorer (usually on the left), select the demo.html file.
6. Review the code responsible for a simple map.
7. In the bottom of the editor, click the Go Live link to launch the html page in a browser.

```
require([
  "esri/Map",
  "esri/views/MapView"
], function(Map, MapView) {

    var map = new Map({
      basemap: "streets"
    });

    var view = new MapView({
      container: "viewDiv",
      map: map,
      zoom: 14,
      center: [-76.925, 40.245]

    });

});
```

Add Basemap Toggle

Allow the user to switch from a streets map to a satellite imagery map.

1. In the explorer on the left, select the 1-basemapToggleSnippet.txt file.
2. Follow the steps copying the code snippets to the demo.html.

```
  "esri/widgets/BasemapToggle",
], function(Map, MapView, BasemapToggle) {
```

```
    var toggle = new BasemapToggle({
      titleVisible: true,
      view: view
      });
    view.ui.add(toggle, "bottom-right");
```

Dark Basemap

Change the default basemap to a dark, nighttime map.

1. In the explorer on the left, select the 1-basemapToggleSnippet.txt file.
2. Follow the steps copying the code snippets to the demo.html.

```
var map = new Map({
            basemap: "streets-night-vector"
        });
```

Add Layer List

Add a layer list to the map which allows the user to turn on/off the layers.

1. In the explorer on the left, select the 3-layerListSnippet.txt file.
2. Follow the steps copying the code snippets to demo.html.

```
"esri/widgets/LayerList",
```

```
    var layer List = new LayerList({
        view: view
      });
    view.ui.add(layerList, "top-right");
```

Add Graphics Layer Setup Code

Add several functions to process GeoJSON files and convert the contents to graphics layers.

1. In the explorer, select the 4-geoJsonLayerSetupSnippet.txt file.
2. Follow the steps copying the code snippets to demo.html.
3. Examine the code contents. The code snippet contains four methods for requesting GeoJSON files, converting the features to graphics, and adding the graphics to a layer.

```
function addGeoJSONLayer(fileName, title, colorArray, visible)
…
function getPointEsriGraphicsFromGeoJson(geoJson, wkid, colorArray)
…
function getLineEsriGraphicsFromGeoJson(geoJson, wkid, colorArray) {
…
function getGraphicDefinition(feature, geometry, symbol) {
…
```

Add the GeoJSON Layers

Add a layer for both the point and line GeoJSON files created previously.

1. In the explorer, select the 5-addGeoJsonLayersSnippet.txt file.
2. Follow the steps copying the code snippets to demo.html.

```
  addGeoJSONLayer("lines.geojson", "Route", [200, 0, 0, 1], false);
  addGeoJSONLayer("points.geojson", "Street Lights", [200, 200, 0, 1], false);
```

Adding Locate Indicator

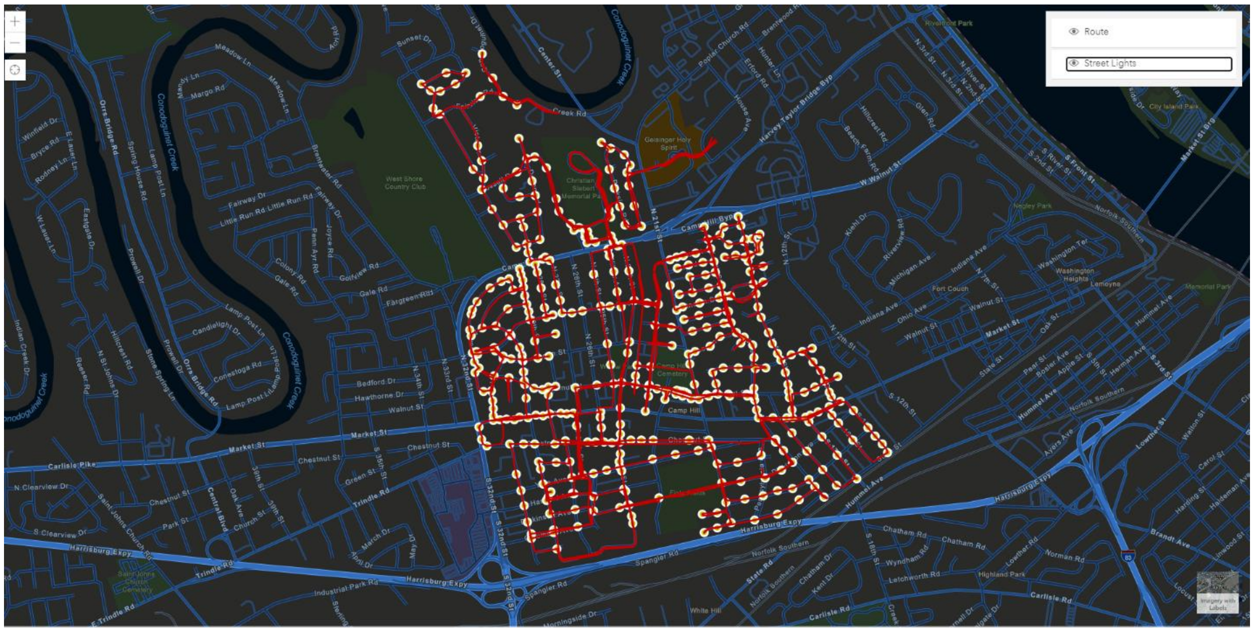
Add a locate indicator to the map.

3. In the explorer on the left, select the 6-locateSnippet.txt file.
4. Follow the steps copying the code snippets to the demo.html

```
    "esri/widgets/Locate",
    var locateBtn = new Locate({
      view: view
    });
    view.ui.add(locateBtn, "top-left");
```

Finished Demo Map

Your finished map should contain a basemap toggle, a layer list, two GeoJSON layers, and a locate button.


Route
Street Lights
Imagery with Labels