Client-Side Web GIS

Leaflet JavaScript API

Image from NASA: https://commons.wikimedia.org/wiki/File:Earth_Western_Hemisphere_transparent_background.png#filelinks

WV VIEW

AmericaView
Empowering Earth Observation Education
AmericaView.org - Est. 2003

1

Background

2

What is Leaflet?

- Web mapping JavaScript API
- Free and open-source
- Used by a variety of organizations:
  - NPR
  - Data.gov
  - Facebook
  - Pinterest, Etsy, Flickr
  - USA Today, The Washington Post
  - European Commission
  - Open Street Map
  - US Forest Service

Leaflet

an open-source JavaScript library
for mobile-friendly interactive maps

https://leafletjs.com/

3

Preparing to use Leaflet

- Download Leaflet JavaScript and CSS libraries and store in your website directory (in “src” folder and “css” folder)
- Link to Content Delivery Network (CDN)
- You do not need an API key to use Leaflet

```
<link rel="stylesheet" href=“css/leaflet.css">

<script src="src/leaflet-src.js"></script>
```

4

4

JavaScript Classes

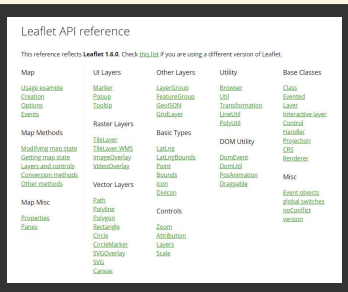
- Blueprint for objects of the same type
- Use constructor method to create a new object of this class
- Pre-defined methods, properties, and options to work with for:
  - Creation
  - Interaction
  - Events
  - Get, set, and change settings and options

5

5

Documentation

- Documentation available with API
- Some examples used here came from the API documentation



Leaflet API reference

This reference reflects **Leaflet 1.6.0**. Check this list if you are using a different version of Leaflet.

Map

Usage example
Creation
Options
Events

Map Methods

Modifying map state
Getting map state
Layers and controls
Conversion methods
Other methods

Map Misc

Properties
Panes

UI Layers

Marker
Popup
Tooltip

Raster Layers

TileLayer
TileLayer.WMS
ImageOverlay
VideoOverlay

Vector Layers

Path
Polyline
Polygon
Rectangle
Circle
CircleMarker
SVGOverlay
SVG
Canvas

Other Layers

LayerGroup
FeatureGroup
GeoJSON
GridLayer

Basic Types

LatLng
LatLngBounds
Point
Bounds
Icon
DivIcon

Controls

Zoom
Attribution
Layers
Scale

Utility

Browser
Util
Transformation
LineUtil
PolyUtil

DOM Utility

DomEvent
DomUtil
PosAnimation
Draggable

Base Classes

Class
Evented
Layer
Interactive layer
Control
Handler
Projection
CRS
Renderer

Misc

Event objects
global switches
noConflict
version

https://leafletjs.com/reference-1.6.0.html

6

6

Document Ready

```
$(document).ready
(function(){
  });
```

7

Map Class

8

Generate a Map

- L.map()

```
map = L.map('formap', {center:[37.1, -95.7],
zoom:4});
```

9

Controls

- AttributionControl = used to provide attribution/citation for basemaps and data layers (this is a legal obligation for using some data)
- zoomControl = add a default zoom controller to the map (will be added by default)

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
  attribution: '&copy; <a href="https://www.openstreetmap.org
  /copyright">OpenStreetMap</a> contributors'
```

10

10

Map State and Interaction

- center = set center of map view
- zoom = set map zoom level
- minZoom = set minimum allowed zoom level
- maxZoom = set maximum allowed zoom level
- Layers = Add layers to the map
- maxBounds = set maximum bonds allowed
- dragging = define whether or not the map can be panned or dragged

11

11

Map Methods

- addControl() = add control to map
- removeControl() = remove control from map
- addLayer() = add layer to map
- removeLayer() = remove layer from map
- setView() = set view using lat/long and zoom level
- setZoom() = set just zoom
- fitBounds() = fit map to bounds (generally relative to a data layer)

12

12

Setting Center and Zoom

```
mymap = L.map('formap', {center:
[37.1, -95.7], zoom: 4});
```

13

Limiting Pan and Zoom

- Set a minimum zoom level with minZoom option
- Set a maximum zoom level with maxZoom option
- Limit panning using maxBounds option
- Not allow dragging or panning using dragging option

14

Working with LatLng

- Represent geographic points
- Provide latitude and longitude as an array or as an object
- Provide latitude before longitude when providing an array

```
var latlng1 = L.latLng(50.5, 30.5);
```

15

Working with LatLngBounds

- Defines a rectangular extent
- Can be used to set map extent

```
var corner1 = L.latLng(40.712, -74.227),
corner2 = L.latLng(40.774, -74.125),
bounds = L.latLngBounds(corner1, corner2);
```

```
map.fitBounds([
    [40.712, -74.227],
    [40.774, -74.125]
]);
```

16

16

Working with Layers

17

17

Layer Types

- User Interface Layers
  - Marker
  - Popup
  - Tooltip
- Raster Layers
  - Tile Layer
  - TileLayer.WMS
  - ImageOverlay
  - VideoOverlay
- Vector Layers
  - Path
  - Polyline
  - Polygon
  - Rectangle
  - Circle
  - CircleMarker
  - SVGOverlay
  - SVG
  - Canvas
- Other Layers
  - LayerGroup
  - FeatureGroup
  - GeoJSON
  - GridLayer

18

18

Panes and zIndex

- The relative horizontal position of layers or and layer groups is defined by assigning a zIndex
- You can also generate our own panes to store layers and content and define your own zIndex for each pane

Pane	Type	Z-index	Description
mapPane	HTMLElement	'auto'	Pane that contains all other map panes
tilePane	HTMLElement	200	Pane for GridLayers and TileLayers
overlayPane	HTMLElement	400	Pane for vectors (Paths, like Polylines and Polygons), ImageOverlays and VideoOverlays
shadowPane	HTMLElement	500	Pane for overlay shadows (e.g. Marker shadows)
markerPane	HTMLElement	600	Pane for Icons of Markers
tooltipPane	HTMLElement	650	Pane for Tooltips.
popupPane	HTMLElement	700	Pane for Popups.

19

Panes and zIndex

```
hpmap.createPane('pA');
hpmap.createPane('pB');
hpmap.createPane('pC');
hpmap.createPane('pD');
hpmap.createPane('pE');
```

```
hpmap.getPane('pA').style.zIndex = 210;
hpmap.getPane('pB').style.zIndex = 410;
hpmap.getPane('pC').style.zIndex = 420;
hpmap.getPane('pD').style.zIndex = 430;
hpmap.getPane('pE').style.zIndex = 440;
```

20

Marker

- Show geographic point as an icon
- L.marker()
- Options
  - Icon
  - Opacity
  - riseOnHover
  - Pane
  - Attribution

```
L.marker([50.5, 30.5]).addTo(map);
```

21

Marker Methods

- getLatLng/setLatLng() = get or set the coordinate of the marker
- getIcon()/setIcon() = get or set the icon used to represent the location
- setOpacity() = change the opacity of the icon
- toGeoJSON() = convert the icon to GeoJSON

22

Add and Style Marker

- Set location
- Define icon to use (default to blue tear drop icon)
- Can use custom icons or icons from libraries

```
var myIcon = L.icon({
    iconUrl: 'my-icon.png',
    iconSize: [38, 95],
    iconAnchor: [22, 94],
    popupAnchor: [-3, -76],
    shadowUrl: 'my-icon-shadow.png',
    shadowSize: [68, 95],
    shadowAnchor: [22, 94]
});
L.marker([50.505, 30.57], {icon: myIcon}) .addTo(map);
```

23

Raster Layers

- TileLayer = represent raster tile layer data
- TileLayer.WMS = display raster tile layers from a Web Map Service (WMS)
- ImageOverlay = load and display a single image on the map over a given bounds
- VideoOverlay = extends ImageOverlay to allow for video data to be added to map over a given bounds (such as a time series animation)

24

Add Raster Tile Layer

- L.tileLayer() 'http://{s}.somedomain.com/blabla/{z}/{x}/{y}{r}.png'
- Options
  - minZoom
  - maxZoom
  - opacity
  - zIndex
  - bounds
  - pane
  - className

25

25

Add Base Maps

- Base maps are generally added as raster tile layers

```
L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png?{foo}', {foo:
'bar', attribution: 'Map data &copy; <a
href="https://www.openstreetmap.org/">OpenStreetMap</a> contributors,
<a href="https://creativecommons.org/licenses/by-sa/2.0/">CC-BY-
SA</a>'}).addTo(map);
```

26

26

Add Base Maps (leaflet-providers)

- Leaflet-providers plugin offers an easy means to add a variety of different base maps

```
var hpmap;
var lyrOSM;
var lyrTer;
var lyrNL;
var lyrDrk;
```

```
lyrOSM = L.tileLayer.provider('OpenStreetMap.Mapnik');
lyrTer = L.tileLayer.provider('OpenTopoMap');
lyrNL = L.tileLayer.provider('NASAGIBS.ViirsEarthAtNight2012');
lyrDrk = L.tileLayer.provider('CartoDB.DarkMatter');
hpmap.addLayer(lyrOSM);
```

https://leaflet-extras.github.io/leaflet-providers/preview/

27

27

Circle

- L.circle()
- Options
  - Radius (radius defined in meters so symbol does not change size with map scale)
  - Stroke, color, weight, opacity, lineCap, lineJoin, dashArray, dashOffset, fill, fillColor, fillOpacity, renderer, className, pane, attribution

28

CircleMarker

- L.circleMarker
- Options
  - Radius (with radius defined in pixels so symbol does change with map scale)
  - Stroke, color, weight, opacity, lineCap, lineJoin, dashArray, dashOffset, fill, fillColor, fillOpacity, renderer, className, pane, attribution

29

Add and Style CircleMarker

```
var cMrk1 = L.circleMarker(latlng, {radius:10,
weight: 1, color: '#111111',
fillColor:"#eeeeee", fillOpacity:0.9,
pane:'pD'});
```

30

Polyline

- L.polyline()
- Accepts array of arrays of latitude and longitude coordinates
- Can use a multidimensional array to represent MutliPolyline shape
- Options: stroke, color, weight, opacity, lineCap, lineJoin, dashArray, dashOffset, fill, fillColor, fillOpacity, fillRule, renderer, className

```
var latlngs = [
  [[45.51, -122.68],
  [37.77, -122.43],
  [34.04, -118.2]],
  [[40.78, -73.91],
  [41.83, -87.62],
  [32.76, -96.72]]
];
```

```
var polyline = L.polyline(latlngs, {color: 'blue'}).addTo(map);
```

31

31

Polygons

- L.polygon()
- Accepts array of arrays of latitude and longitude coordinates
- First and last should not be the same coordinate
- Can use multidimensional arrays to represent MultiPolygon shape
- Can include holes

```
var latlngs = [
  [[37, -109.05],[41, -109.03],[41, -102.05],[37, -102.04]], // outer ring
  [[37.29, -108.58],[40.71, -108.58],[40.71, -102.50],[37.29, -102.50]] // hole
];
```

32

32

Polygon Styling

- Options: stroke, color, weight, opacity, lineCap, lineJoin, dashArray, dashOffset, fill, fillColor, fillOpacity, fillRule, renderer, className

```
var latlngs = [[37, -109.05],[41, -109.03],[41, -102.05],[37, -102.04]];
var polygon = L.polygon(latlngs, {color: 'red'}).addTo(map);
```

33

33

LayerGroup

- Group layers to handle as a single layer
- L.layerGroup()

```
L.layerGroup([marker1, marker2])
  .addLayer(polyline)
  .addTo(map);
```

34

34

FeatureGroup

- Extends layer group
- Makes it easier to do the same thing to all members of the group

```
L.featureGroup([marker1, marker2, polyline])
  .bindPopup('Hello world!')
  .on('click', function() { alert('Clicked on a
      member of the group!'); })
  .addTo(map);
```

35

35

GeoJSON

- Link to GeoJSON via a URL or stored within your website directory (for example, in "data" folder)
- Rendered using similar methods as those for makers, circleMarkers, circles, polylines, and polygons
- Extends featureGroup
- Use a function to define how GeoJSON spawns leaflet layers

36

36

Reading in GeoJSON

- L.geoJSON() or $.getJSON()
- L.geoJson.ajax() (from leaflet-ajax plugin)
  - Call JSON using AJAX or JSONP

```
lyrpfor = L.geoJSON.ajax('data/per_for_wgs84.geojson',
{style:stylePfor, onEachFeature:popPfor})

lyrcities = L.geoJSON.ajax('data/cities_wgs84.geojson',
{pointToLayer:returnCtyStyle).addTo(hpmap);
```

37

37

Styling GeoJSON

- Similar to Leaflet layers
- Can use a function

```
lyrstates = L.geoJSON.ajax('data/states_wgs84.geojson',
{style:{fillColor:'#ffffff', color:"red", fillOpacity:0,
opacity:1, weight:1.5, pane:'pB'}}).addTo(hpmap);
```

38

38

Styling GeoJSON

- Similar to Leaflet layers
- Can use a function

```
function stylePfor(json){
   var att = json.properties;
   if (att.wester_158 <= 2) {
      var clrPfor = '#edf8fb';
   } else if (att.wester_158 > 2 && att.wester_158 <= 5){
      var clrPfor = '#b2e2e2';
   } else if (att.wester_158 > 5 && att.wester_158 <= 10){
      var clrPfor = '#66c2a4';
   } else if (att.wester_158 > 10 && att.wester_158 <= 15){
      var clrPfor = '#2ca25f';
   } else {
      var clrPfor = '#006d2c';
   }
   return {fillColor: clrPfor, opacity:1, color:"#222222",
    fillOpacity:.9, weight:1, pane:'pC'}
}
```

39

39

Styling GeoJSON

```
function returnCtyStyle(json, latlng){
    var att = json.properties;
    switch (att.POP_CLASS) {
      case 5:
        var optCty = {radius:2, weight: 1, color: '#111111', fillColor:"#eeeeee", fillOpacity:0.9, pane:'pD'};
        break;
      case 6:
        var optCty = {radius:4, weight: 1, color: '#111111', fillColor:"#eeeeee", fillOpacity:0.9, pane:'pD'};
        break;
      case 7:
        var optCty = {radius:6, weight: 1, color: '#111111', fillColor:"#eeeeee", fillOpacity:0.9, pane:'pD'};
        break;
      case 8:
        var optCty = {radius:8, weight: 1, color: '#111111', fillColor:"#eeeeee", fillOpacity:0.9, pane:'pD'};
        break;
      case 9:
        var cMrk1 = L.circleMarker(latlng, {radius:10, weight: 1, color: '#111111', fillColor:"#eeeeee", fillOpacity:0.9, pane:'pD'};
        break;
    }
    var popupOptions = {maxWidth: 200, pane:'pE', className: 'popC'};
    return L.circleMarker(latlng, optCty);
}
```

40

40

Working with ESRI Data Layers

- ESRI Leaflet plugin allows you to use ESRI data layers in Leaflet
- Can connect to layers made available through a data service, stored locally with your webpage, or hosted on ArcGIS Online
- Can connect to: ESRI Basemaps, ESRI Vector Basemaps, feature layers, hosted feature layers, raster tile layers, dynamic raster tile layers, map layers, and image map layers
- Allow for generation of symbology and pop-ups

http://esri.github.io/esri-leaflet/

41

41

Working with ESRI Data Layers

```
layer = L.esri.basemapLayer('Topographic').addTo(wvmap);
naip = L.esri.tiledMapLayer({url: '#'}).addTo(wvmap);
inter = L.esri.featureLayer({url: '#', style:{color: "orange", weight:
3}}).addTo(wvmap);
```

http://esri.github.io/esri-leaflet/

42

42

Controls

43

43

Zoom and Pan Control

- Leaflet has default zoom controls
- Can add different zoom controls with plugins
  - Example: Leaflet.zoomslider (http://kartena.github.io/Leaflet.zoomslider/)
  - Example: L.Control.ZoomBar (https://github.com/elrobis/L.Control.ZoomBar)
  - Example: Leaflet.twofingerzoom (https://github.com/aratcliffe/Leaflet.twofingerzoom)
  - Example: Leaflet.BoxZoom (https://github.com/gregallensworth/L.Control.BoxZoom)
- No pan control provide by base Leaflet so must use plug-ins
  - Example: Leaflet.Pancontrol (http://kartena.github.io/Leaflet.Pancontrol/)
  - Example: Leaflet.BorderPan (https://github.com/slara/Leaflet.BorderPan)

44

44

Layer Controls

- Can use as a base map switcher
- Can use to turn layers on and off

```
ctlLayers =
L.control.layers(objBasemaps,
objOper).addTo(mymap);
```

```
objBasemaps = {
    "OSM": lyrOSM,
    "OSM Terrain":lyrTer,
    "Night Lights":lyrNL,
    "Dark":lyrDrk
};

objOper = {
   "Cities":lyrcities,
   "Percent Forest": lyrpfor,
   "Mean Annual Temp.":lyrtemp,
   "Total Annual Precip.":lyrprecip,
   "States":lyrstates
};
```

45

45

Scale Control

- L.control.scale()
- Options:
  - Maxwidth
  - Metric
  - Imperial
  - undateWhenIdle
  - position

```
L.control.scale().addTo(map);
```

46

46

Pop-ups and Tool Tips

47

47

Pop-Ups vs. Tooltips

- Generate a pop-up tied to a map feature or coordinate
- Tooltip generally used for text only

48

48

Build Pop-Ups

- Use .bindPopup() or L.popup()
- Options: maxWidth, minWidth, maxHeight, minHeight, keepInView, closeButton, className
- Content specified using HTML and attribute data
- Style with CSS using className option

```
L.circleMarker(latlng, optCty).bindPopup("<p><b>" + att.NAME + ", "
+ att.ST + "</b><br>Elevation: " + att.elev + " <br>Annual Precip: " +
att.precip + "</p>", popupOptions);
```

49

49

Build Tooltip

- Similar to methods for pop-up

```
return L.marker(latlng, optCty).bindTooltip("Font-
Awesome Icon");
```

50

50

Handling Events

51

51

Common Events

- Change view extent
- Change map center
- Change zoom level
- Add/remove a layer
- Change layer symbology
- Pan to a location

52

52

Set/Change Center and Zoom

- Example using jQuery
- Other useful map methods: setZoom(), zoomIn(), zoomOut(), fitBounds(), fitWorld(), panTo(), flyTo()

```
$("#btnSalz").click(function(){
    pmap.setView([47.798,13.059], zoom=12)
});
```

53

53

Add and Remove Layers

- Useful methods: addLayer(), removeLayer(), hasLayer(), eachLayer()

```
$("#btnAddRem").click(function(){
    if (pmap.hasLayer(lyrPhotos) == true) {
        pmap.removeLayer(lyrPhotos)
    } else {
        pmap.addLayer(lyrPhotos)
});
```

54

54

Change Symbology or Style

```
$("#btnMrk2").click(function(){
  pmap.removeLayer(lyrPhotos);
   lyrPhotos = L.geoJSON.ajax('data/photos_points.geojson',
  {pointToLayer:returnPntStyle1}).addTo(pmap);
});
```

55

55

Easy Button

- Plugin that allows you to easily add a button to a Leaflet map and set up an action or an event for it to handle
- Can use custom icons or icons from a library
- L.EasyButton = https://github.com/CliffCloud/Leaflet.EasyButton

```
ctlEasybutton = L.easyButton('fa-compress-alt', function(){
                    ctlSidebar.toggle();
                }).addTo(pmap);
```

56

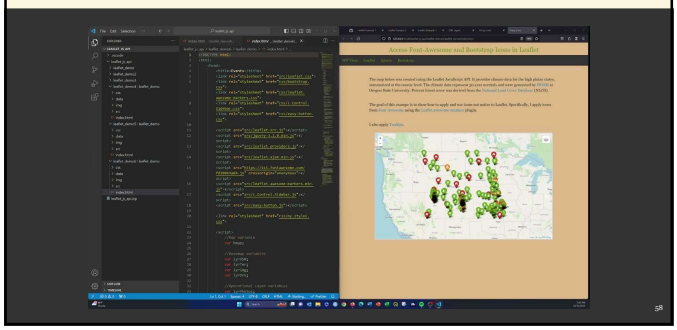
56

Examples

57

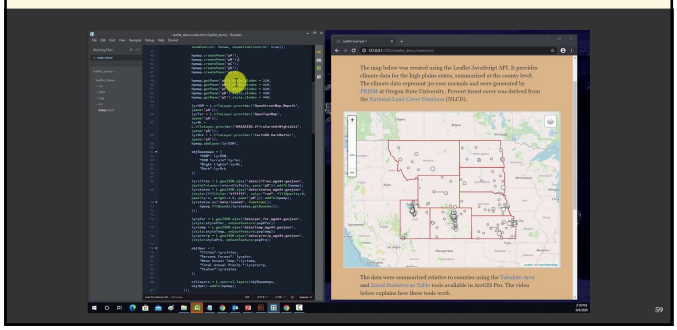
57

Video: Leaflet Examples Overview



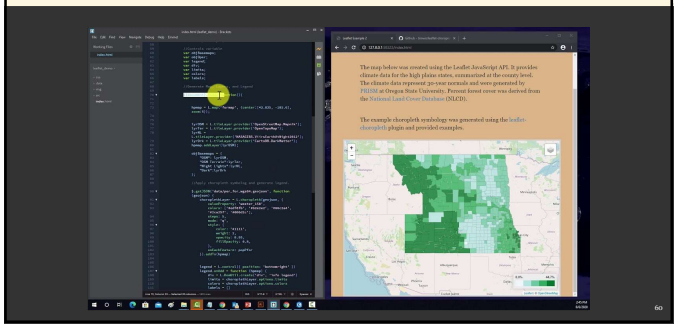
58

Example 1



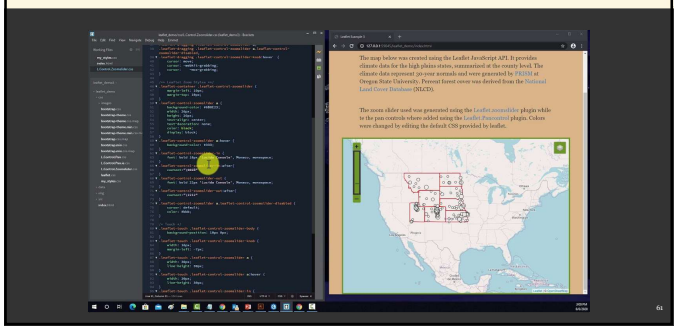
59

Example 2



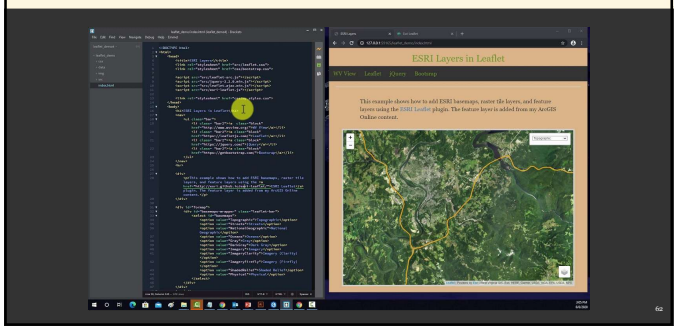
60

Example 3



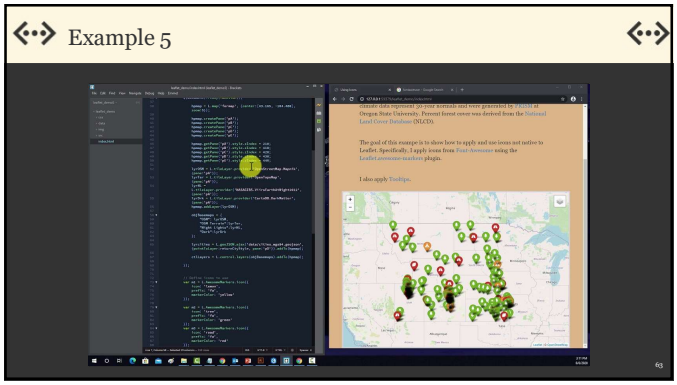
61

Example 4



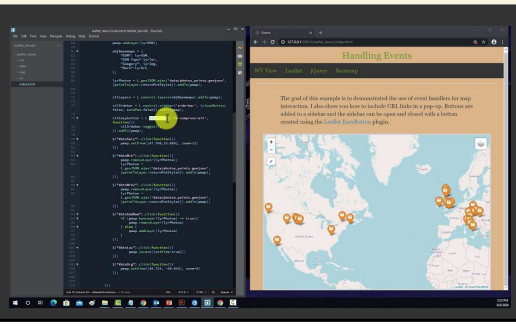
62

Example 5



63

Example 6



64

This is the end of this lecture module.

Please return to the West Virginia View Webpage for additional content.

65