Leaflet 入門

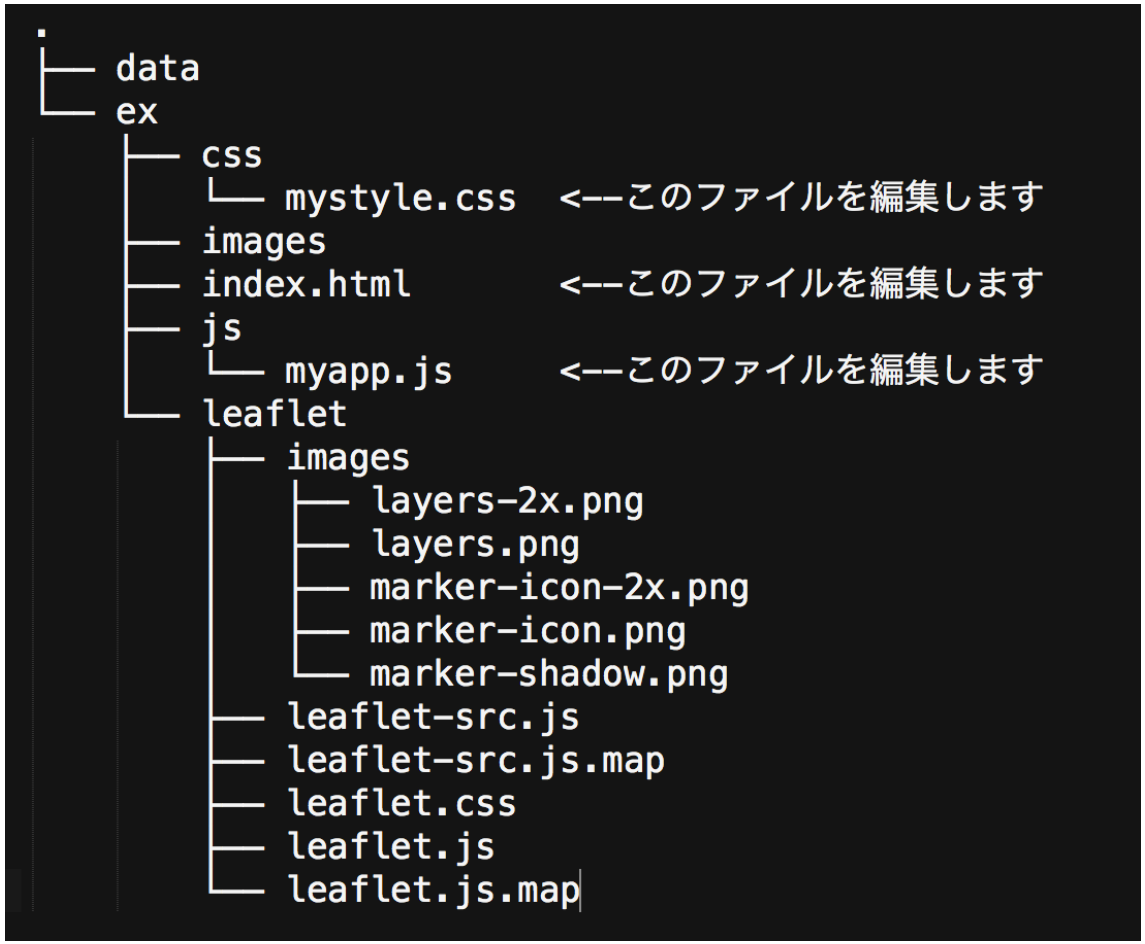
Leafletとは

- JavaScriptのWebマップ API
- 最新バージョンは2017年8月リリースのv1.2.0（2017年11月現在）
- 軽量（約38KB）
- 滑らな動作
- デスクトップ、モバイルの双方で動作
- 公式サイト http://leafletjs.com/

このハンズオンでやること

- ベースの地図を表示する
- 地図上にLayerを表示する
  - Marker, Polyline, Polygon, FeatureLayer, GeoJSON
- 地図上にControlを表示する
- インタラクティブな地図にする
- プラグインを使ってみる

ファイル構成


.
data
ex
css
mystyle.css <--このファイルを編集します
images
index.html <--このファイルを編集します
js
myapp.js <--このファイルを編集します
leaflet
images
layers-2x.png
layers.png
marker-icon-2x.png
marker-icon.png
marker-shadow.png
leaflet-src.js
leaflet-src.js.map
leaflet.css
leaflet.js
leaflet.js.map

ベースの地図を表示する

ベースの地図を表示する

Leafletでは、ラスタータイルを使うのが基本

ラスタータイルとは…

1枚の正方形の画像で全世界（注）を表せる縮尺をズーム0として…

注: webメルカトル図法では北極南極を表現できない等、図法によって表示範囲に制限がある

ラスタータイルとは…

ズーム1では縦2 * 横2の4枚

ラスタータイルとは…

ズーム2では縦4＊横4の16枚

ラスタータイルとは…

縦横に＜2 のズーム値乗＞枚の画像をタイル状に配置したもの

ズーム0

ズーム1

ズーム2

…ズームn

ベースの地図を表示する

index.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Leaflet Hands On</title>
    <link rel="stylesheet" href="./leaflet/leaflet.css"/>
    <link rel="stylesheet" href="./css/mystyle.css">
    <script src="./leaflet/leaflet.js"></script>
</head>
<body>
    <h1>この画面いっぱいに地図を表示してみましょう。</h1>
    <script src="./js/myapp.js"></script>
</body>
</html>
```

ベースの地図を表示する

./index.html

```
<div id="map"></div>
```

./css/mystyle.css

```
body{
    padding:0;
    margin:0;
}

#map{
    width:100vw;
    height:100vh;
}
```

mapオブジェクトの定義

```
// L.Mapクラスのインスタンス
var map = new L.Map('map');

// こちらがよく使われる
var map = L.map('map');
```

ベースの地図を表示する

ラスターレイヤーの定義

```
// syntax
// var tile_layer = L.tileLayer(url,options);


// Open Street Map
var osm_baselayer =  L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
     attribution: 'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
});


// Open Street Map Japan
var osm_j_baselayer = L.tileLayer('http://tile.openstreetmap.jp/{z}/{x}/{y}.png',{
    attribution:
       'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
});


// 国土地理院 地理院タイル 標準地図
var gis_baselayer = L.tileLayer('https://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.png', {
    attribution:
 '<a href="http://www.gsi.go.jp/kikakuchousei/kikakuchousei40182.html “target="_blank">国土地理院</a>'
});
```

ベースの地図を表示する

ベースの地図を表示するまでの一連の流れ

```
var map = new L.Map('map');

// Open Street Map Japan
var osm_j_baselayer = L.tileLayer('http://tile.openstreetmap.jp/{z}/{x}/{y}.png',{
    attribution:
    'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
});

osm_j_baselayer.addTo(map);
map.setView([35.1611,136.8842], 16);
```

ベースの地図を表示する

こういう書き方もできる

```
var map = new L.Map('map').setView([35.1611,136.8842], 16);

// Open Street Map Japan
var osm_j_baselayer = L.tileLayer('http://tile.openstreetmap.jp/{z}/{x}/{y}.png',{
    attribution:
    'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map);
```

こういう書き方もできる

```
// Open Street Map Japan
var osm_j_baselayer = L.tileLayer('http://tile.openstreetmap.jp/{z}/{x}/{y}.png',{
    attribution:
    'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
});

// setting all as options
var map = new L.Map('map',{
    center:[35.1611,136.8842],
    zoom:16,
    layers:[osm_j_baselayer]
});
```

ベースの地図を表示する

緯度経度の指定方法

```
// Leafletでは、下記は全て同様に扱われる
var latLng = new L.LatLng(35.1611, 136.8842);

var latLng = L.latLng(35.1611, 136.8842);

var latLng = {lat:35.1611, lng:136.8842};

var latLng = [35.1611, 136.8842];
```

地図上にLayerを表示する

LayerとControl

Leafletでの定義

- Layer：地図を動かした時に、地図と一緒に動くもの
- Control：地図を動かしても、画面の同じ位置に表示され続けるもの

Marker （点データ）

Marker （点データ）

```
// syntax
//L.marker(latLng,options?)


// Markerを定義して、地図に表示
var my_marker = L.marker([35.1611,136.8842]).addTo(map);


// これも同じこと
var my_marker = L.marker([35.1611,136.8842]);
map.addLayer(my_marker);
```

Marker （点データ）

ポップアップや、ツールチップをつけることができる

```
// popup
my_marker.bindPopup("HELLO, <a href=\"http://leafletjs.com\">Leaflet</a> !”);


// tooltip
my_marker.bindTooltip("HELLO Leaflet!");
```

Marker （点データ）

任意の画像をアイコンに設定できる

```
// custom image icon
var cross_icon = L.icon({
    iconUrl:'./images/cross.gif',
    iconRetinaUrl:'./images/cross@2x.gif',
    iconSize:[40,40],
    iconAnchor:[20,20],
    popupAnchor:[0,0]
});

var my_marker = L.marker([35.1611,136.8842],{
    icon:cross_icon,
});
```

Marker （点データ）

Html要素をアイコンに設定できる

```
// custom image icon
var div_icon = L.divIcon({
    className:"my-div-icon",
    html:"<div>↑ 愛知大学</div><div>名古屋キャンパス</div>",
    iconSize:[100,40],
    iconAnchor:[0,0]
});


var my_marker = L.marker([35.1611,136.8842],{
    icon:div_icon,
});
```

./css/mystyle.css

```
.my-div-icon{
    background-color:#289728;
    color:#ffffff;
    border-radius: 8px;
    padding:5px;
}
```

Polyline （線データ）

Polyline （線データ）

```
// syntax
// L.polyline(coordinates,options?);


// coordinatesはL.LatLngの配列
var route_to_aichi_univ_coords = [
    [35.170931759646294, 136.88297510147098],
    [35.17131764533583, 136.884241104126],
    [35.169072466563655, 136.88484191894534],
    [35.1676691983501, 136.88518524169925],
    [35.16463454803477, 136.8855714797974],
    [35.163967962276885, 136.885678768158],
    [35.162389184741635, 136.88608646392825],
    [35.16212605217294, 136.8855929374695],
    [35.16195062998756, 136.8845629692078],
    [35.1617401228657, 136.88381195068362],
    [35.161494530534924, 136.88396215438846]
];

var my_route = L.polyline(route_to_aichi_univ_coords).addTo(map);
```

Polyline （線データ）

スタイルを設定できる

```
var route = L.polyline(route_to_aichi_univ_coords,{
    color:”#ff0000”,
    weight:6,
    dashArray:"20, 10, 5,10",
    lineCap:"square",
    lineJoin:"square"
}).addTo(map);
```

Polygon（面データ）

Polygon（面データ）

```
// syntax
// L.polygon(coordinates,options?);

// simple polygon coordinates
var aichi_univ_coords = [
    [35.16138050527214, 136.8836295604706],
    [35.16163486910046, 136.88459515571597],
    [35.16141558998541, 136.88472390174869],
    [35.16137173409146, 136.8846380710602],
    [35.16124016626777, 136.88472390174869],
    [35.161266479849516, 136.8848526477814],
    [35.16103842852472, 136.884959936142],
    [35.16074020659675, 136.88390851020816]
];

var aichi_univ = L.polygon(aichi_univ_coords).addTo(map);
```

Polygon（面データ）

穴あきのPolygon

```
// outer and inner rings
var polygon_with_holes_coords = [
    // outer ring
    [
        [35.16030164295172, 136.88329696655276],
        [35.1604244410107, 136.88585042953494],
        [35.15809124619016, 136.8861293792725],
        [35.15788072908032, 136.8834471702576]
    ],
    // inner rings
    [
        [
            [35.159792906161144, 136.88391923904422],
            [35.15977536345644, 136.88452005386355],
            [35.159266623374364, 136.8845629692078],
            [35.15928416618878, 136.8837904930115]
        ],
        [
            [35.158915766291535, 136.8844556808472],
            [35.15898593782917, 136.88539981842044],
            [35.1583719348225, 136.88548564910892],
            [35.15830176275521, 136.8846917152405]
        ]
    ]
];

var polygon_with_holes = L.polygon(polygon_with_holes_coords).addTo(map);
```

Polygon（面データ）

スタイルを設定できる

```
var aichi_univ = L.polygon(aichi_univ_coords,{
    color:"#dc143c",
    fillColor:"#800000",
    fillOpacity:0.8,
    dashArray:"20, 10, 5,10",
}).addTo(map);
```

.getBounds()

緯度経度の範囲（L.LatLngBounds）を取得するメソッド

```
// returns bounds of current view
var map_bounds = map.getBounds();

// polyline has bounds.
var my_route_bounds = my_route.getBounds()
// polygon has bounds.
var aichi_univ_bounds = aichi_univ.getBounds()
```

表示緯度経度の範囲を地図にセットすることができる

```
var north_east = [35.1604244410107, 136.8861293792725],
south_west= [35.15788072908032, 136.88329696655276],
bounds = [south_west,north_east];

map.fitBounds(bounds);
```

Layerを一つのまとまりとして扱う

Layerを一つのまとまりとして扱う

L.LayerGroupと L.FeatureGroup

```
// syntax
// L.featureGroup(<Layer[]> layers)

// FeatureGroupを定義して、地図に表示
var overlays = L.featureGroup([my_marker,my_route,aichi_univ]).addTo(map);


// 後からLayerを追加できる
polygon_with_holes.addTo(overlays);
```

Layerを一つのまとまりとして扱う

全Layerの緯度経度範囲を取得できる

```
var marker_group = L.layerGroup([marker1,marker2,marker3]);
var marker_bounds = marker_group.getBoudns();
```

全Layerをまとめたイベント（後述）扱える（注）

```
var marker_group = L.featureGroup([marker1,marker2,marker3])
.addTo(map)
.on("click",function(e){
    // do something
});
```

注: L.LayerGroupはイベントを発火しない。イベントを扱う場合はL.FeatureGroupを使う。

Layerを一つのまとまりとして扱う

L.Control.Layersでまとめて表示／非表示の切り替えができる

```
// syntax
// L.control.layers(<Object> baseLayers?, <Object> overlays?)

L.control.layers(
    {"OSM Japan":osm_j_baselayer},
    {"my overlays":overlays})
.addTo(map);
```

GeoJSON

GeoJSON

./index.html

```
    <div id="map"></h1>
    <script src="../data/geojson.js"></script>
    <script src="./js/myapp.js"></script>
```

```
// syntax
// L.geoJSON(<Object> geojson?, <Object> options?)

var nursery_homes = L.geoJSON(hoikusho_data).addTo(map);
```

GeoJSON

Layerの生成をフックしてカスタマイズ

```
// setting DivIcon
var nursery_homes = L.geoJSON(hoikusho_data,{
    onEachFeature:function(feature, layer){
        var icon = L.divIcon({
            html:feature.properties.name + "<br>"+feature.properties.address,
            iconSize:[200,50]
        });
        layer.setIcon(icon);
    }
}).addTo(map);
```

GeoJSON

フィルタリング

```
// filtering nursery homes with name
var nursery_homes = L.geoJSON(hoikusho_data,{
    onEachFeature:function(feature, layer){
        var icon = L.divIcon({
            html:feature.properties.name + "<br>"+feature.properties.address,
            iconSize:[200,50]
        });
        layer.setIcon(icon);
    },
    filter:function(feature){
        if (feature.properties.name.match(/.*愛.*/)){
            return true;
        }
        return false;
    }
}).addTo(overlays);
```

GeoJSON

個々のデータの属性値によって表示を変える

```
// 地価公示最高額
var max_price=8850000;

var decimalToHex = function(decimal_value){
    const values = "0123456789abcdef";

    var hex_value = "";
    for (var i=0;decimal_value > 0; i++){
        var decimal_for_digit = decimal_value % (16 ** (i + 1));
        var index_for_digit = decimal_for_digit / (16 ** i);
        hex_value = values[index_for_digit] + hex_value;
        decimal_value = decimal_value - decimal_for_digit;
    }
   return hex_value;
};

var land_prices = L.geoJSON(h25_chika_kouji_data,{
    pointToLayer:function(feature, latLng){
        var red_value = Math.floor(255 * (feature.properties.price/max_price));
        var blue_value = 255 - red_value;
        var red_str = decimalToHex(red_value);
        var blue_str = decimalToHex(blue_value);
        return L.circleMarker(latLng,{
            radius:feature.properties.price * 0.000005,
            color:"#"+ red_str + "00" + blue_str,
            fillColor:"#"+ red_str + "00" + blue_str,
            fillOpacity:1
        });
    }
}).addTo(map);
```

Control

Control

Scale

```
L.control.scale({
    metric:true,
    imperial:false,
    position:"bottomleft"
}).addTo(map);
```

Control

ZoomとAttributionはデフォルトで表示されている

```
// disabling zoom and attribution control
var map = L.map('map',{zoomControl:false,attributionControl:false});

L.control.zoom({position:"bottomright"}).addTo(map);

L.control.attribution({position:"topleft"}).addTo(map);
```

Control

独自のControlを定義

```
L.Control.MyControlClass = L.Control.extend({
    onAdd:function(the_map){
        var div = L.DomUtil.create("div");
        div.setAttribute("class","my-custom-control");
        div.innerHTML = "MY CUSTOM Control";
        L.DomEvent.on(div,"click",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"dblclick",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"mousedown",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"mousewheel",L.DomEvent.stopPropagation);

        return div;
    },
    onRemove:function(the_map){

    }
});

var my_control = new L.Control.MyControlClass({position:"bottomleft"});
my_control.addTo(map);
```

インタラクティブな地図にする

インタラクティブな地図にする

イベントハンドラの定義

```
map_or_layer.on(event_name, function(e){
        // do something ...
},context);
```

インタラクティブな地図にする

clickイベント

```
var nursery_homes = L.geoJSON(h25_chika_kouji,{
    onEachFeature(feature, layer){

        // clickで、ポップアップに属性を表示

        layer.on("click",function(e){
            var content =  “name: " + feature.properties.name
                        +  “<br>price: " + feature.properties.price
            this.bindPopup(content).openPopup();
        },layer);
    }
}).addTo(map);
```

インタラクティブな地図にする

contextmenu（右click）イベント

```
var nursery_homes = L.geoJSON(h25_chika_kouji,{
    onEachFeature(feature, layer){

        // clickで、ポップアップに属性を表示

        layer.on("click",function(e){
            var content =  "name:" + feature.properties.name
                        +  "<br>price:Â¥" + feature.properties.price
            this.bindPopup(content).openPopup();
        },layer)

        // 右clickで、削除

        .on("contextmenu",function(e){
            this.removeFrom(nursery_homes);
        },layer);
    }
}).addTo(map);
```

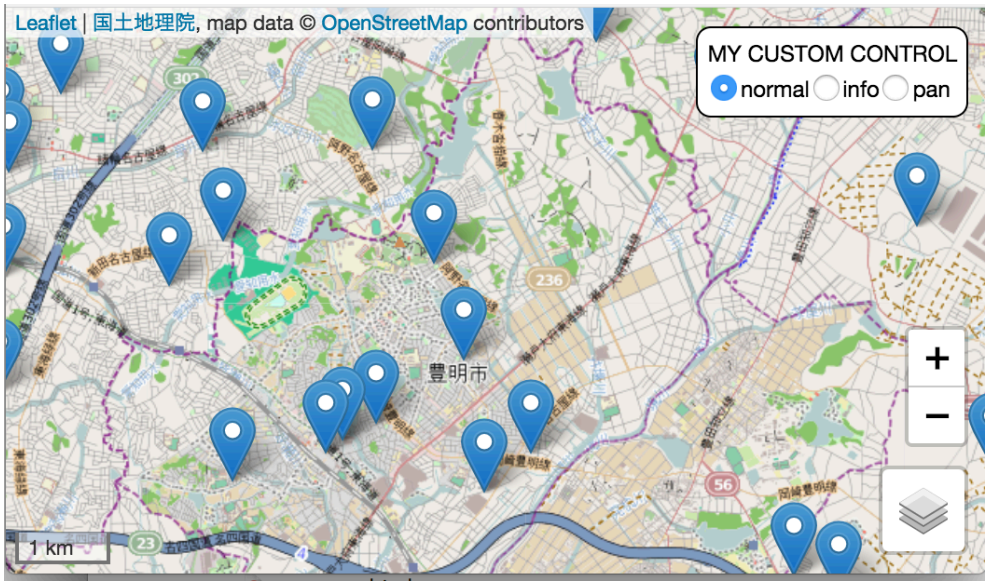
インタラクティブな地図にする

イベントハンドラの解除

```
map_or_layer.off(event_name, event_handler);
```

インタラクティブな地図にする

イベントハンドラを切り替えるカスタムControl



インタラクティブな地図にする

イベントハンドラを切り替えるカスタムControl

```
L.Control.MyControlClass = L.Control.extend({
    onAdd:function(the_map){
        var div = L.DomUtil.create("div");
        div.setAttribute("class","my-custom-control");
        div.innerHTML = "MY CUSTOM CONTROL";
        L.DomEvent.on(div,"click",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"dblclick",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"mousedown",L.DomEvent.stopPropagation);
        L.DomEvent.on(div,"mousewheel",L.DomEvent.stopPropagation);

        var mode_selector_container = L.DomUtil.create("form");
        div.append(mode_selector_container);
        var mode_normal = L.DomUtil.create("input");
        mode_normal.setAttribute("type","radio");
        mode_normal.setAttribute("name","mode");
        mode_normal.setAttribute("value","normal");
        mode_normal.checked = true;
        mode_selector_container.append(mode_normal);
        mode_selector_container.append("normal");

        var mode_info = L.DomUtil.create("input");
        mode_info.setAttribute("type","radio");
        mode_info.setAttribute("name","mode");
        mode_info.setAttribute("value","info");
        mode_selector_container.append(mode_info);
        mode_selector_container.append("info");

        var mode_pan = L.DomUtil.create("input");
        mode_pan.setAttribute("type","radio");
        mode_pan.setAttribute("name","mode");
        mode_pan.setAttribute("value","pan");
        mode_selector_container.append(mode_pan);
        mode_selector_container.append("pan");
```

次のスライドに続く

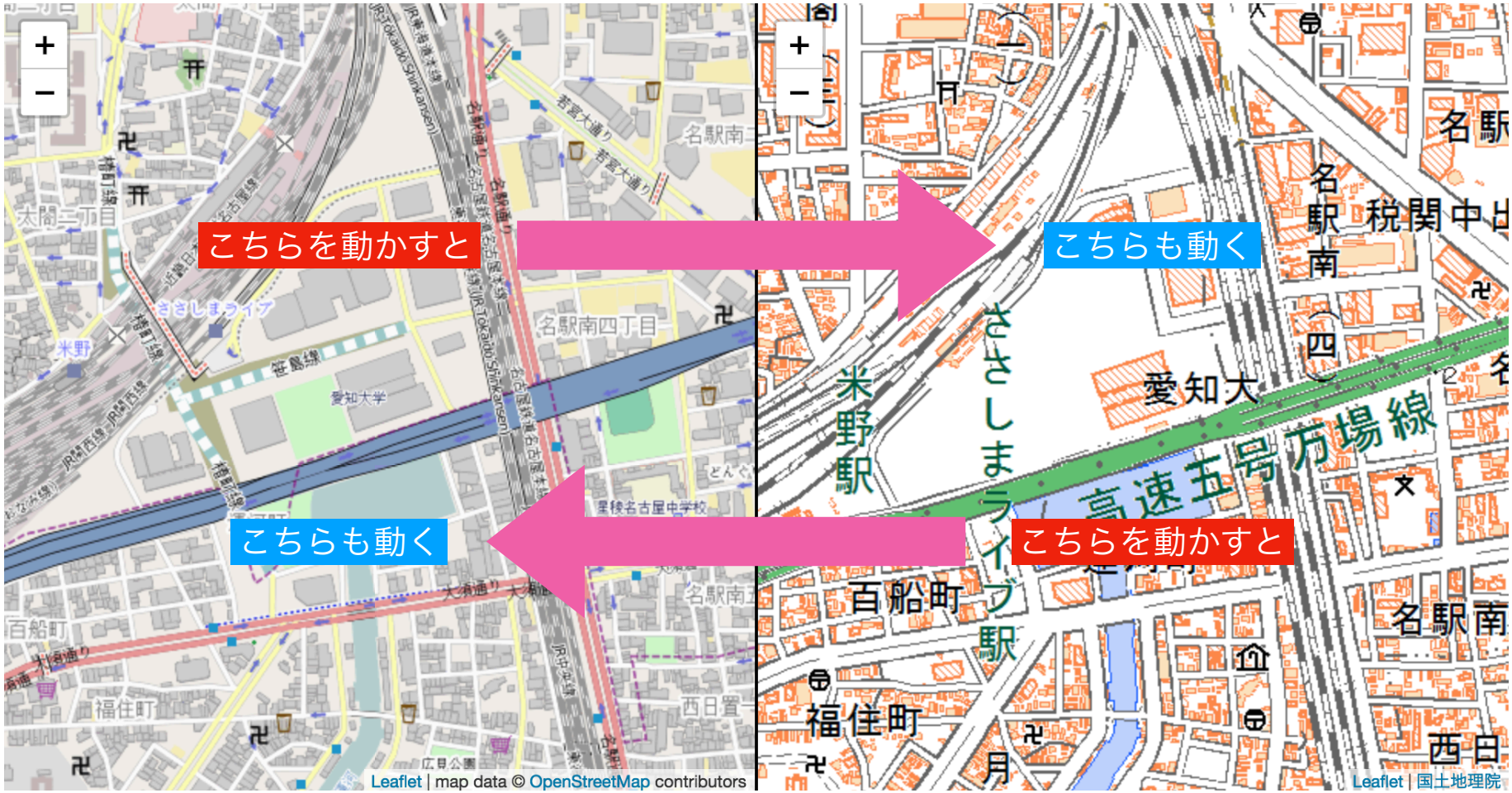
続き

```
                const click_handlers = {
                    "info":function(e){
                        var latLng = e.latlng;
                        map.openPopup(latLng.toString(),latLng);
                    },
                    "pan":function(e){
                        map.panTo(e.latlng);
                    },
                }
                this.click_handler = null;


                L.DomEvent.on(mode_selector_container,"change",function(e){
                    this.clearClickHandler();
                    tool_mode =mode_selector_container.mode.value;
                   if (tool_mode == "normal"){
                        return this;
                    }

                    this.click_handler = click_handlers[tool_mode];
                    map.on("click",this.click_handler);
                    return this;

                },this);

                return div;
            },
            onRemove:function(the_map){
                this.clearClickHandler();
            },
            clearClickHandler:function(){
                if (this.click_handler){
                    map.off("click",this.click_handler);
                    this.click_handler = null;
                }
            }
        });
```

インタラクティブな地図にする

二つの地図を同期させる

完成イメージ



インタラクティブな地図にする

二つの地図を同期させる

./twomaps.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Leaflet Hands On</title>
    <link rel="stylesheet" href="./leaflet/leaflet.css"/>
    <link rel="stylesheet" href="./css/mystyle.css">
    <script src="./leaflet/leaflet.js"></script>
</head>
<body>
    <div id="map1" class="twomaps"></div>
    <div id="map2" class="twomaps"></div>
    <script src="./js/twomaps.js"></script>
</body>
</html>
```

./css/mystyle.css

```
.twomaps{
    width:49vw;
    height:100vh;
    float:left;
    border-right:2px solid #000;
}
```

インタラクティブな地図にする

二つの地図を同期させる

```
var options = {
 center : [35.1611,136.8842],
 zoom : 16
}

var map1 = L.map('map1',options);
var osm_j_baselayer = L.tileLayer('http://tile.openstreetmap.jp/{z}/{x}/{y}.png',{
    attribution:'map data &copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'
}).addTo(map1);

var map2 = L.map('map2',options);
var gis_baselayer = L.tileLayer('https://cyberjapandata.gsi.go.jp/xyz/std/{z}/{x}/{y}.png', {
    attribution:
“<a href=\"http://www.gsi.go.jp/kikakuchousei/kikakuchousei40182.html\" target=\"_blank\">国土地理院</a>"
}).addTo(map2);
```

次のスライドに続く

続き

```
var moveOtherMap= function(target,center,zoom){
    offHandler(target);
    target.setView(center,zoom);
};

var offHandler = function(target){
    target.off("move");
};

var onHandler = function(this_map){
    var other_map;
    if (this_map === map1){
        other_map = map2;
    }else{
        other_map = map1;
    }

    this_map.on("move",function(e){
        moveOtherMap(other_map,this.getCenter(),this.getZoom());
    },this_map);
};

L.DomEvent.on(map1.getContainer(),"mouseover touchstart",function(e){
    onHandler(this);
},map1);

L.DomEvent.on(map2.getContainer(),"mouseover touchstart",function(e){
    onHandler(this);
},map2);
```

プラグインを使ってみる