

ArcGIS API for JavaScript を使用したアプリ開発

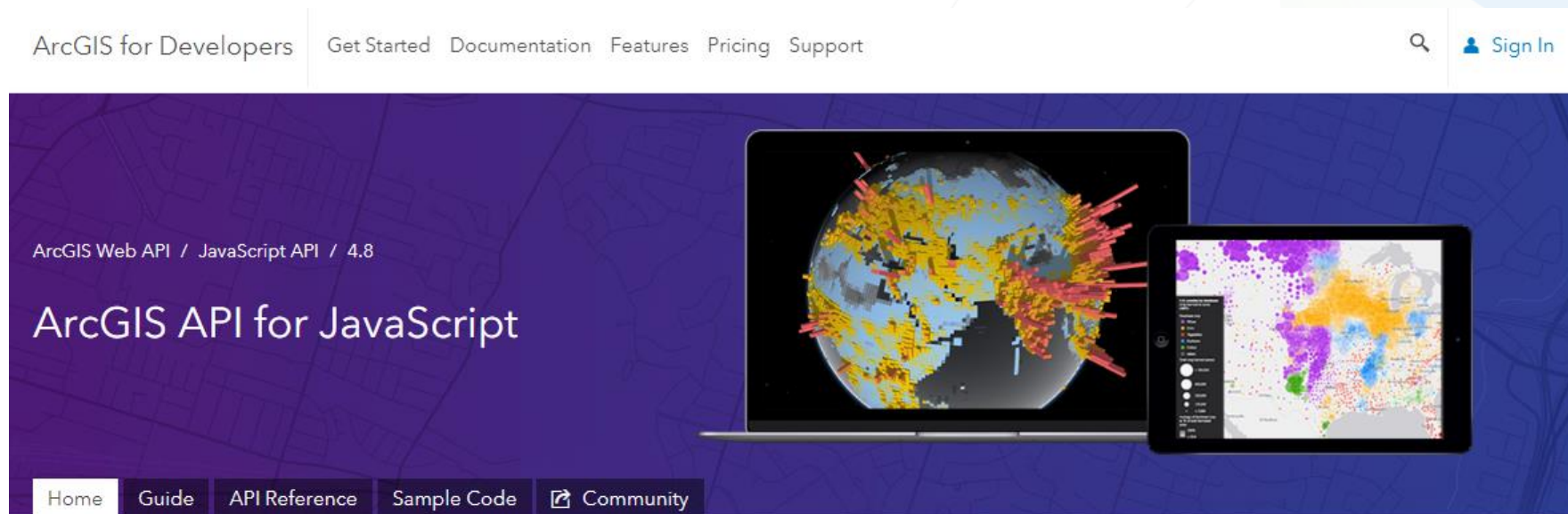
2018/8/24
ESRIジャパン株式会社

目次

- ArcGIS API for JavaScript の導入
 - セットアップ
- ArcGIS API for JavaScript の利用
 - マップの作成
 - レイヤーの操作
 - シンボルとレンダラー
 - マップの探索
 - ポップアップ
 - グラフィックとジオメトリ
 - フィーチャのクエリ
 - ウィジェット

ArcGIS API for JavaScript の導入

開発のスタート



Build 2D and 3D web apps!

The 4.x series of the ArcGIS API for JavaScript integrates 2D and 3D into a single, easy-to-use, powerful mapping API. Create 3D maps as easily as 2D maps, seamlessly integrate your web GIS and build influential data-driven visualizations.

[Learn about 4.8](#)

```
// Reference the JavaScript API from our CDN and you are ready to get started:  
<link rel="stylesheet" href="https://js.arcgis.com/4.8/esri/css/main.css">  
<script src="https://js.arcgis.com/4.8/"></script>
```

Need to build a full-featured 2D web app with capabilities such as editing and support for all existing layer types?

[Go to 3.25](#)

Choosing a version ▶

バージョンの選択

ArcGIS Web API / JavaScript API / 4.8 / Guide

ArcGIS API for JavaScript

Home Guide API Reference Sample Code Community

> Get Started

▼ Migrating from 3.x

Choose a version

Migrating from 3.x to 4.8

3.x to 4.x functionality matrix

> Migrating from other APIs

> Working with the API

> Visualization

> Reference

Choose between version 3.25 and 4.8

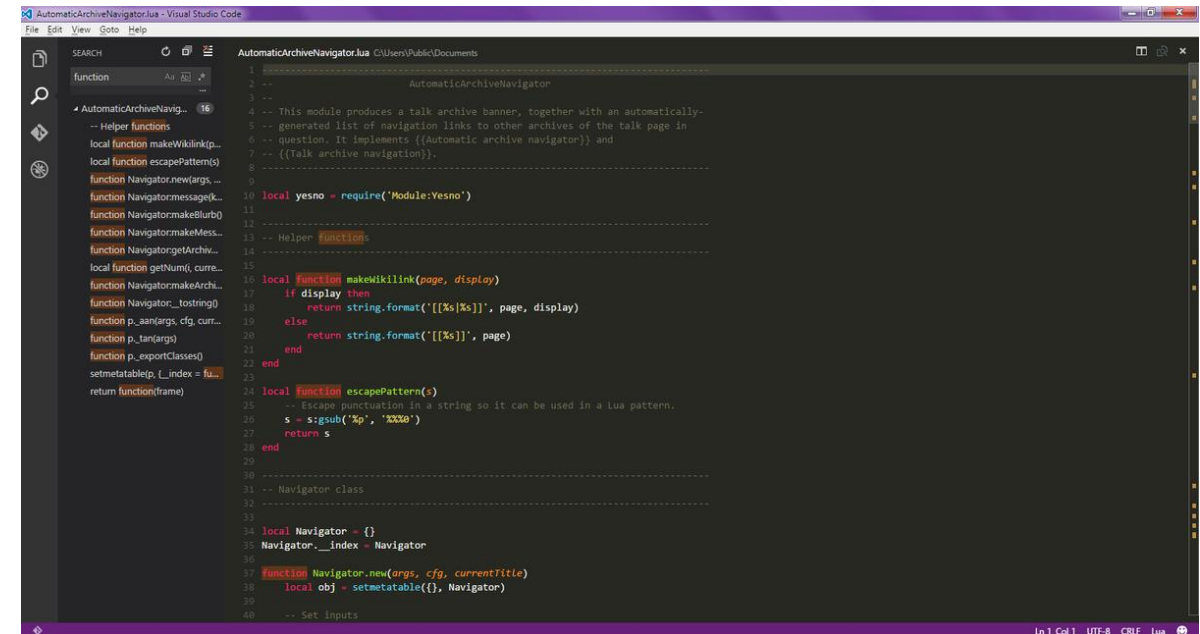
The 4.x API introduces new capabilities such as 3D support, map rotation, and deeper portal integration. However, not all 3.x capabilities are included in the initial 4.x release. Each release will add more functionality until it not only matches 3.x but far exceeds it. Developers should consider their app requirements and evaluate whether the current 4.x or 3.x release has the desired capabilities. For example:

- Does the app need 3D visualization? If so, use 4.x.
- Do you need a particular functionality from 3.x that's not yet available in 4.x such as editing tools or time support? If so, use 3.x.

Capability	3.25	4.8
3D	Not available	Released
2D	Released	Released (partial support)
Vector Tile Layer	Released	Released
Raster Tile Layer	Released	Released
Imagery Layer	Released	Released
Map Image Layer	Released	Released

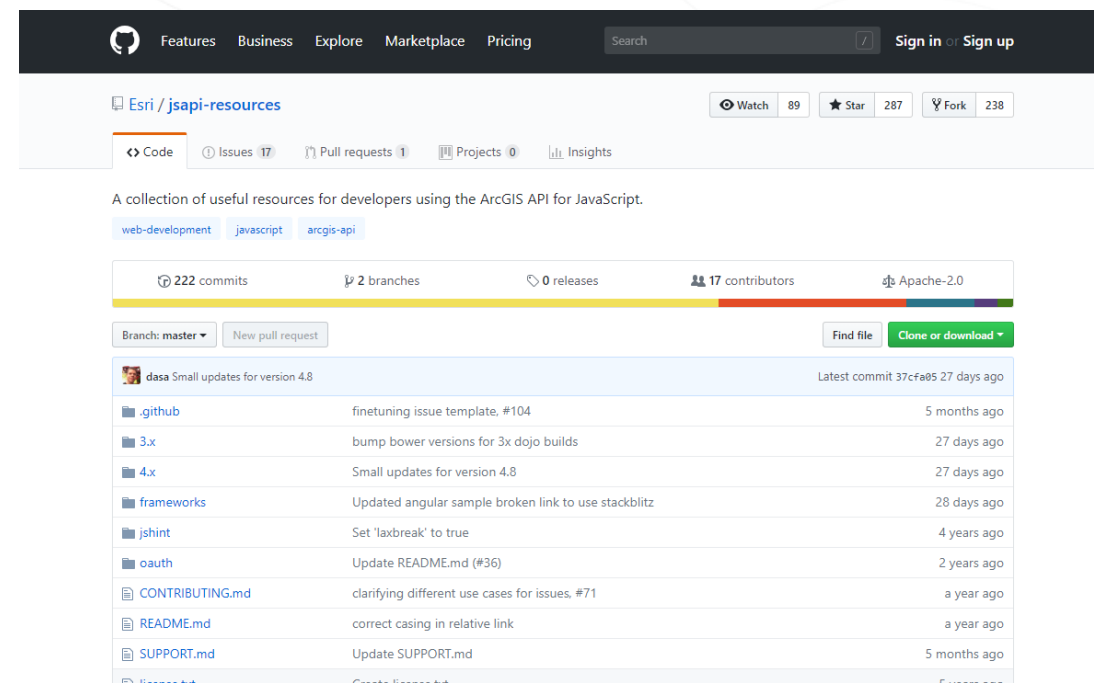
開発環境

- Visual Studio Code
- Sublime Text
- WebStorm
 - など
- 簡単なコードであれば
オンラインエディタでも可能
 - CodePen
 - jsFiddle
 - JS Bin など



JSAPI Resources

- JSHint ファイル
- TypeScript 型定義ファイル
- OAuth ポップアップのコールバック ページ
 - など



参考 : jsapi-resources

<https://github.com/Esri/jsapi-resources>

API の利用

- CDN を利用

- 最も一般的な方法

```
<link rel="stylesheet" href="https://js.arcgis.com/4.8/esri/css/main.css">  
<script src="https://js.arcgis.com/4.8/"></script>
```

- npm を利用

- カスタム ビルドの作成

- ダウンロード して利用

- インターネット環境にアクセスできないネットワーク環境で使う場合など

参考 : Get the API

<https://developers.arcgis.com/javascript/latest/guide/get-api/index.html>

CSS

- **main.css**

- API 全体のスタイルを含んだスタイルシート

```
<link rel="stylesheet" href="https://js.arcgis.com/4.8/esri/css/main.css">
```

- **テーマ**

- **<theme-name>** は light, dark, light-blue, dark-red など設定可能

```
<link rel="stylesheet" href="https://js.arcgis.com/4.8/esri/themes/<theme-name>/main.css">
```

- **カスタム CSS (Sass)**

参考 : Styling

<https://developers.arcgis.com/javascript/latest/guide/styling/index.html>

ArcGIS API for JavaScript の利用

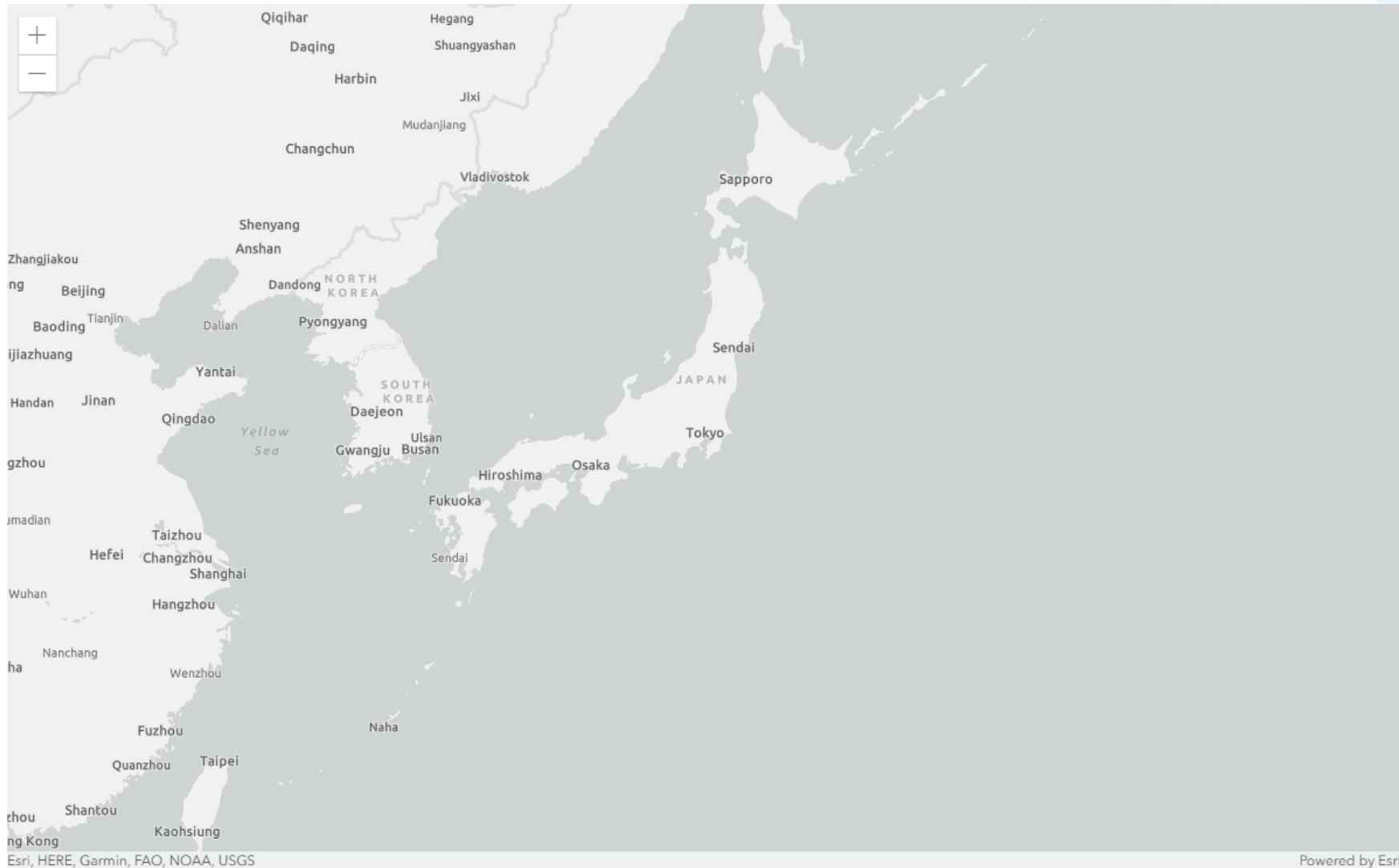
最初のステップ

- どのようにアプリケーションを書くか？
- モジュール単位にファイルを分ける or 1つのファイルに書く？

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <meta name="viewport" content="initial-scale=1,maximum-scale=1,user-scalable=no">
  <title>Step 1: Create a map</title>
  <link rel="stylesheet" href="https://js.arcgis.com/4.8/esri/css/main.css">
  <link rel="stylesheet" href="css/main.css">
  <script src="https://js.arcgis.com/4.8/"></script>
  <script src="js/main.js"></script>
</head>
<body>
  <div id="viewDiv"></div>
</body>
</html>
```

```
└─ Step1_Map
  └─ css
    └─ # main.css
  └─ js
    └─ JS main.js
  └─ <> index.html
```

ステップ1：マップの作成



MapView と SceneView

- Map のデータを可視化

```
var view = new MapView({  
  container: "viewDiv",  
  map: map,  
  zoom: 12,  
  center: [-117.168, 32.776]  
});
```

```
var view = new SceneView({  
  container: "viewDiv",  
  map: map,  
  camera: {  
    heading: 210,  
    tilt: 78,  
    position: {  
      x: -8249335,  
      y: 4832005,  
      z: 50.7,  
      spatialReference: {  
        wkid: 3857  
      }  
    }  
  }  
});
```

Tips : 一般的にはまるポイント

- モジュールの順番が異なる
- モジュールの欠如
- CSS の欠如

後からLegendを加えたが、忘れている

```
require([
  "esri/Map",
  "esri/views/MapView",
  "esri/layers/FeatureLayer",
  "esri/widgets/Legend",
  "dojo/domReady!"
], function(
  Map,
  MapView,
  FeatureLayer
) {
  ~省略~
})
```

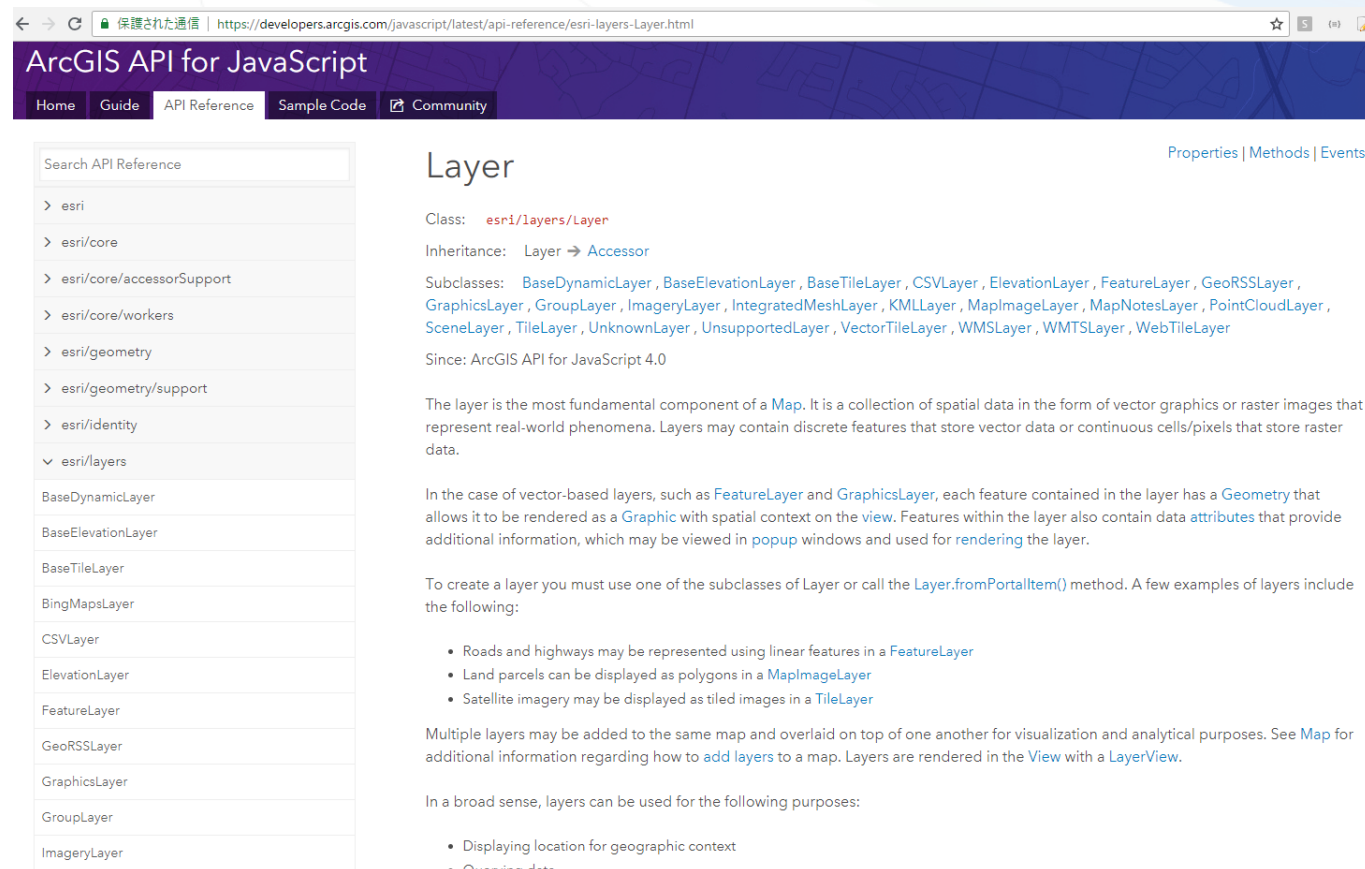


レイヤーの追加

- さまざまなレイヤーのタイプがある

- 基本の追加ステップは同じ

- モジュールのロード
- レイヤーのインスタンス作成
- プロパティのセット
- Map に追加



The screenshot shows the ArcGIS API for JavaScript documentation page for the `Layer` class. The page has a dark blue header with the title "ArcGIS API for JavaScript" and navigation links: Home, Guide, API Reference (selected), Sample Code, and Community. On the right side of the header, there are links for Properties, Methods, and Events. The main content area is divided into two columns. The left column contains a search bar and a tree view of the API reference, with `esri/layers` expanded. The right column contains the details for the `Layer` class, including its inheritance from `Accessor`, a list of subclasses, and a description of the class. The description states that a layer is a collection of spatial data and can contain discrete features or continuous cells/pixels. It also mentions that vector-based layers like `FeatureLayer` and `GraphicsLayer` have a `Geometry` property. The page also includes a section on how to create a layer and a list of examples.

Search API Reference

- > esri
- > esri/core
- > esri/core/accessorSupport
- > esri/core/workers
- > esri/geometry
- > esri/geometry/support
- > esri/identity
- ▼ esri/layers
 - BaseDynamicLayer
 - BaseElevationLayer
 - BaseTileLayer
 - BingMapsLayer
 - CSVLayer
 - ElevationLayer
 - FeatureLayer
 - GeoRSSLayer
 - GraphicsLayer
 - GroupLayer
 - ImageryLayer

Layer

Class: `esri/layers/Layer`

Inheritance: `Layer` → `Accessor`

Subclasses: `BaseDynamicLayer`, `BaseElevationLayer`, `BaseTileLayer`, `CSVLayer`, `ElevationLayer`, `FeatureLayer`, `GeoRSSLayer`, `GraphicsLayer`, `GroupLayer`, `ImageryLayer`, `IntegratedMeshLayer`, `KMLLayer`, `MapImageLayer`, `MapNotesLayer`, `PointCloudLayer`, `SceneLayer`, `TileLayer`, `UnknownLayer`, `UnsupportedLayer`, `VectorTileLayer`, `WMSLayer`, `WMTSLayer`, `WebTileLayer`

Since: ArcGIS API for JavaScript 4.0

The layer is the most fundamental component of a `Map`. It is a collection of spatial data in the form of vector graphics or raster images that represent real-world phenomena. Layers may contain discrete features that store vector data or continuous cells/pixels that store raster data.

In the case of vector-based layers, such as `FeatureLayer` and `GraphicsLayer`, each feature contained in the layer has a `Geometry` that allows it to be rendered as a `Graphic` with spatial context on the `view`. Features within the layer also contain data `attributes` that provide additional information, which may be viewed in `popup` windows and used for `rendering` the layer.

To create a layer you must use one of the subclasses of `Layer` or call the `Layer.fromPortalItem()` method. A few examples of layers include the following:

- Roads and highways may be represented using linear features in a `FeatureLayer`
- Land parcels can be displayed as polygons in a `MapImageLayer`
- Satellite imagery may be displayed as tiled images in a `TileLayer`

Multiple layers may be added to the same map and overlaid on top of one another for visualization and analytical purposes. See `Map` for additional information regarding how to `add layers` to a map. Layers are rendered in the `View` with a `LayerView`.

In a broad sense, layers can be used for the following purposes:

- Displaying location for geographic context
- Quervino data

参考 : Layer

<https://developers.arcgis.com/javascript/latest/api-reference/esri-layers-Layer.html>

Tips : プロパティの設定/取得

- get/set 文を一切必要としない

```
var map = new Map();  
map.basemap = "streets";  
map.ground = "world-elevation";  
var view = new MapView();  
view.center = [-100, 40];  
view.zoom = 6;
```

- コンストラクター内でプロパティを set 可能

```
var map = new Map({  
  basemap: "streets",  
  ground: "world-elevation"  
});  
var view = new MapView({  
  map: map,  
  center: [-100, 40],  
  zoom: 6  
});
```

参考 : Working with properties

Tips : プロパティの監視

- プロパティの変更を Watch で監視

```
layer.watch("loadStatus", function(status) { // do something });
```

- esri/core/watchUtils のユーティリティ メソッドも利用可能

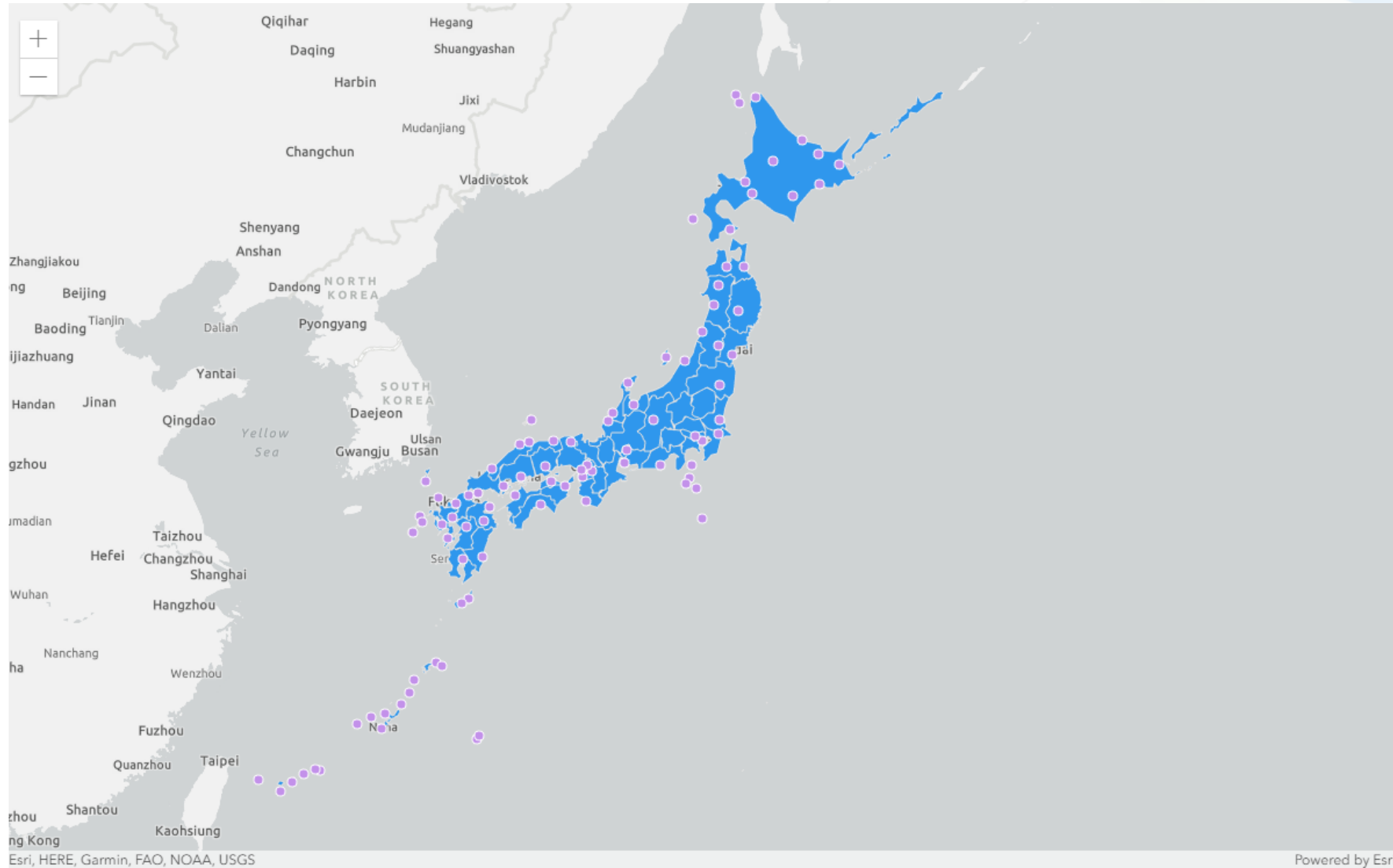
- Watch for changes サンプル

- プロパティを監視し、変更を表示

参考 : Watch for changes

<https://developers.arcgis.com/javascript/latest/sample-code/watch-for-changes/index.html>

ステップ2：レイヤーの追加



レンダラー

- レイヤーで使うシンボルセットを定義
- シンボルの使用方法のルールを設定
- 基本的なコーディングパターン

```
var layerRenderer = new UniqueValueRenderer(); // レンダラーの設定
var featurelayer = new FeatureLayer({
  url: "featurelayer url",
  renderer: layerRenderer // フィーチャ レイヤーに作成したレンダラーを設定
})
```

参考 : Renderer

<https://developers.arcgis.com/javascript/latest/api-reference/esri-renderers-Renderer.html>

シンボル

- レンダラーはシンボルを使用する
 - ポイント、ライン、ポリゴン
- レンダラーのシンボルを設定

```
var symbol = new SimpleMarkerSymbol({  
  // プロパティの設定  
});
```


```
var renderer = new UniqueValueRenderer({  
  defaultSymbol: symbol, // レンダラーへシンボルを設定  
  // その他の必要なプロパティを定義  
});
```

参考 : Symbol

<https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-Symbol.html>

Tips : Autocasting

- モジュールを利用するのに require() する必要がない
- リファレンス内の autocast タグを検索



```
renderer Renderer autocast
```

- Create a layer from portal item サンプル
 - autocasting を利用してレイヤーを作成

参考 : Create a layer from a portal item

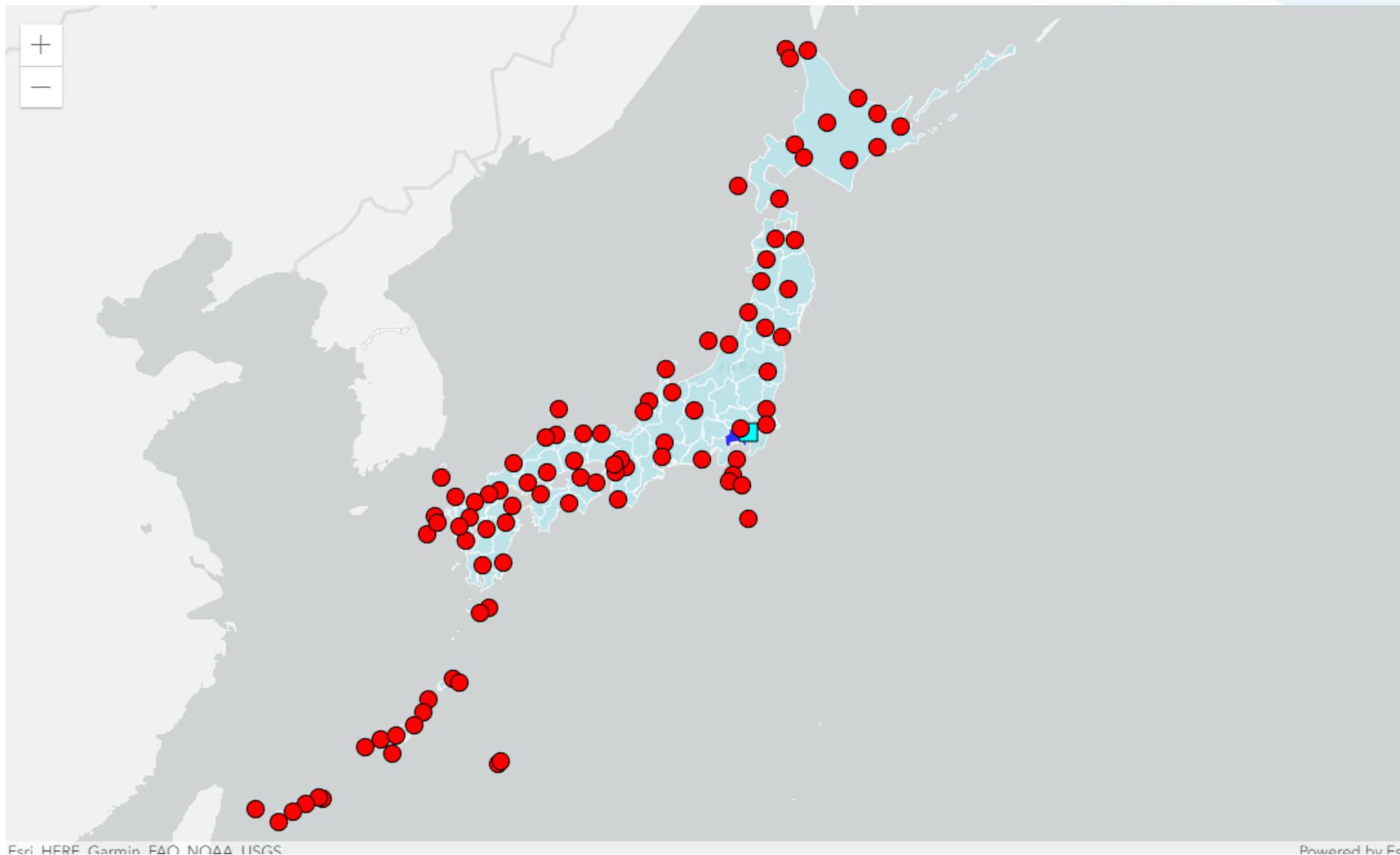
<https://developers.arcgis.com/javascript/latest/sample-code/layers-portal/index.html>

- Autocasting ガイド

参考 : Autocasting

<https://developers.arcgis.com/javascript/latest/guide/autocasting/index.html>

ステップ3：フィーチャ レイヤーのレンダラーの更新



ポップアップの表示

- マウス クリックに応答
- 情報の提供
 - フィーチャの属性
 - 位置
 - 検索結果
- カスタマイズ可能

ポップアップ テンプレート

- View に関連付けられた Popup にコンテンツを設定
- FeatureLayer は popupTemplate プロパティへコンテンツを定義
- ポップアップの表示位置は dockOption を利用して制御

```
var popupTemplate = new PopupTemplate({  
  title: "Title of the popup",  
  content: [{  
    // コンテンツの設定  
  }]  
});
```


ステップ4：ポップアップの追加



データのクエリ

- **FeatureLayer の条件式 (definitionExpression) を定義**
 - Where 句を使用してクライアント側でフィーチャをフィルタリング
 - 大規模なデータを扱うときに有効

```
select.addEventListener("change", function(e) {  
    var uniqueVal = select.value;  
    var expr = uniqueVal === "" ? "" : "ESTAB_MANAGE = " + uniqueVal + "";  
    airportPoint.definitionExpression = expr;  
    ...  
});
```

- **FeatureLayer のクエリ**
 - 検索対象のフィーチャを取得

```
var query = airportPoint.createQuery();  
query.where = "ESTAB_MANAGE = " + uniqueVal + "";  
airportPoint.queryFeatures(query).then(function(results) {  
    ...  
});
```

グラフィックとジオメトリ

- 地点やクエリ結果などの一時的なジオメトリを描画

- ジオメトリ
- シンボル
- 属性
- ポップアップ テンプレート

- グラフィックス レイヤーへ追加

- Intro to graphics サンプル

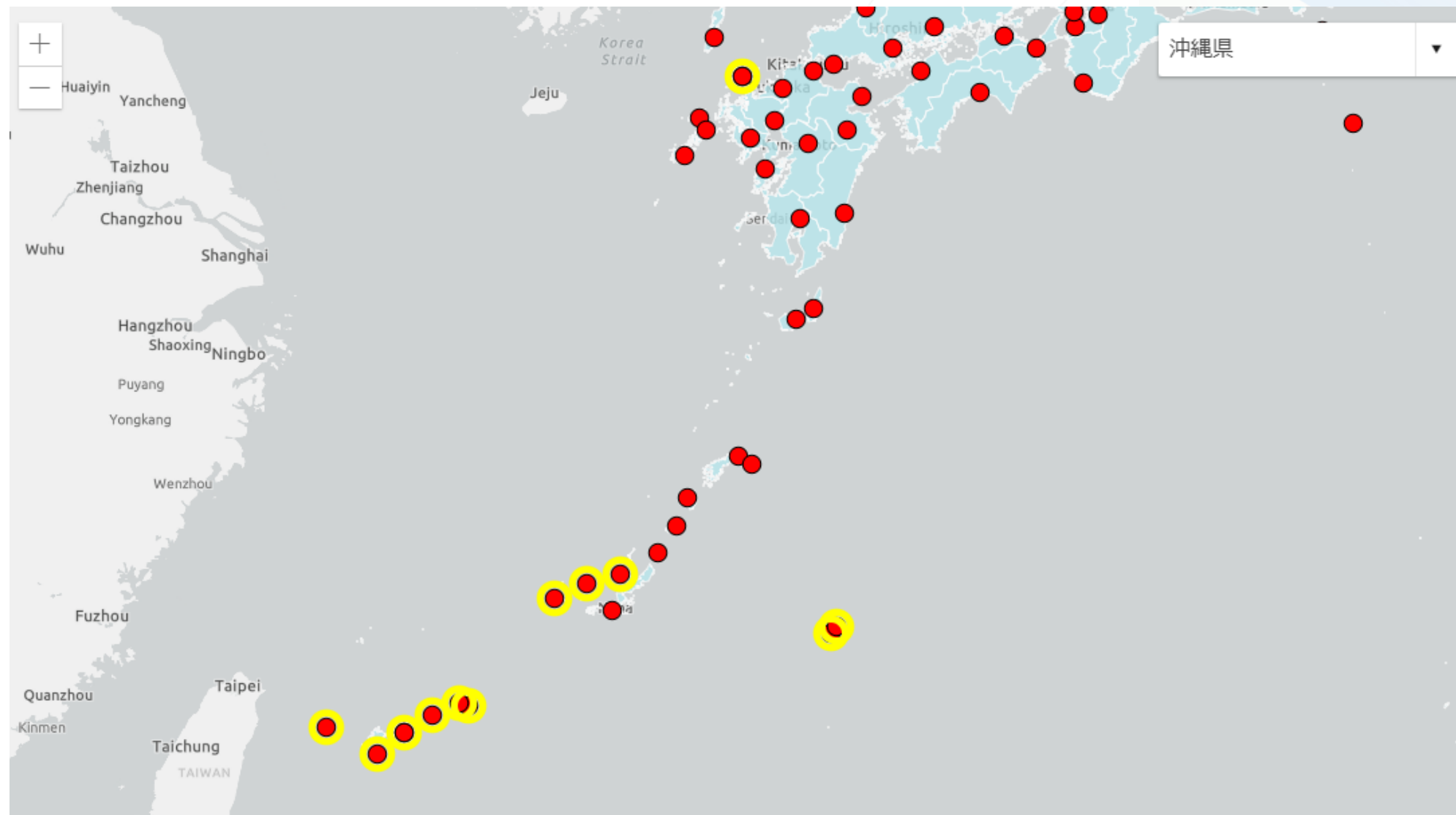
- ポイント、ライン、ポリゴンのグラフィックを作成、表示

```
var graphic = new Graphic({
  geometry: {
    type: "point", // new Point() の autocasts
    ...
  },
  symbol: {
    type: "simple-marker", // new SimpleMarkerSymbol() の autocasts
    ...
  },
  attributes: {
    ...
  },
  popupTemplate: {
    ...
  }
});
```

参考 : Intro to graphics

<https://developers.arcgis.com/javascript/latest/sample-code/intro-graphics/index.html>

ステップ5：フィーチャのクエリ

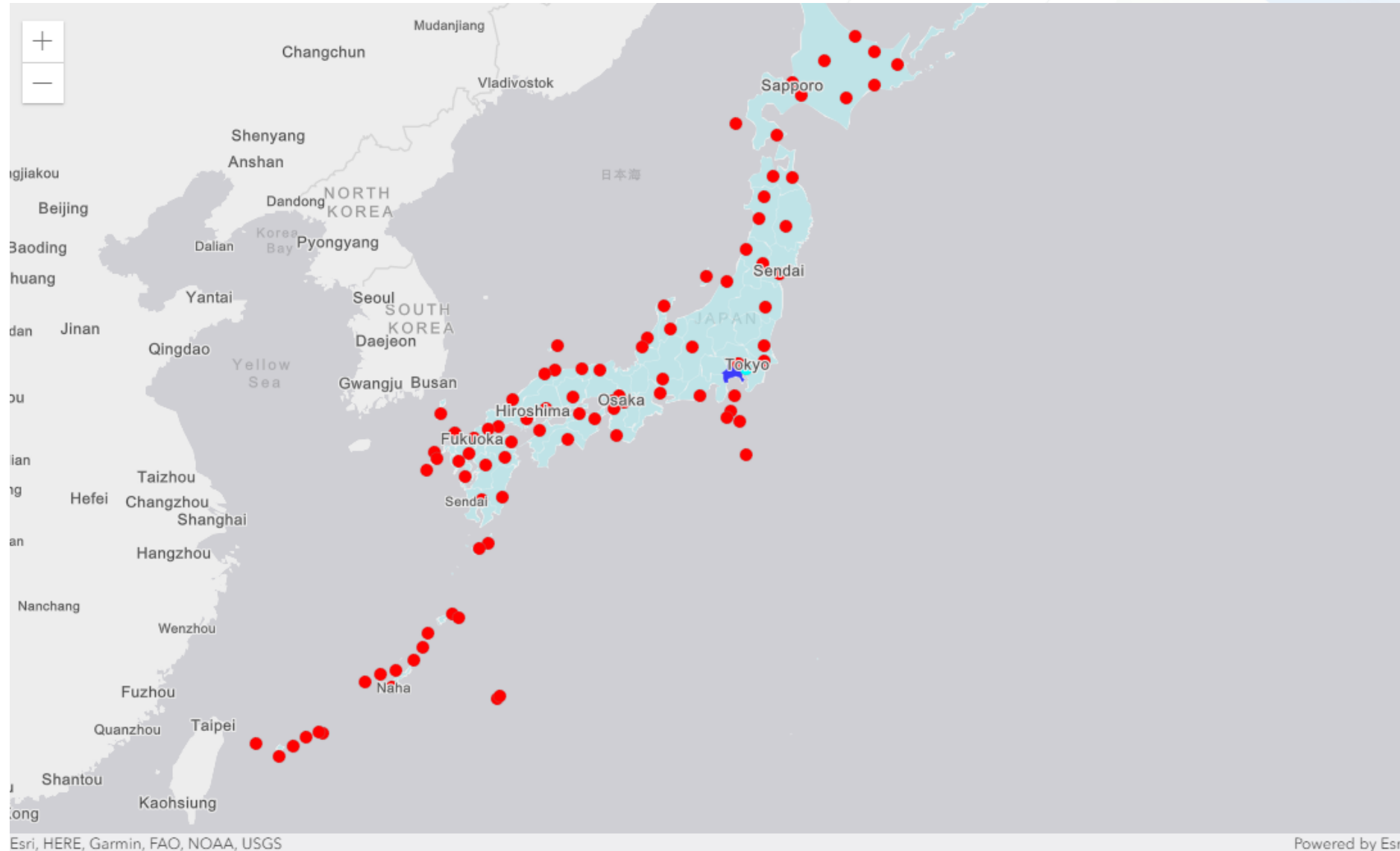


Web マップの利用

- コーディングの労力を軽減
- レンダリングやポップアップなどの全てのカスタマイズを保持

```
var map = new WebMap({  
  portalItem: {  
    id: "8444e275037549c1acab02d2626daaee" // portalItem の autocast  
  }  
});
```

ステップ6 : Web マップの利用



ウィジェット

- 機能をカプセル化
- 全てウィジェットで同様のコーディングパターン

```
view.when(function){  
  var featurelayer = map.layers.getItemAt(1);  
  // 1. ウィジェットの作成  
  var legend = new Legend({  
    // 2. プロパティを設定  
    view: view,  
    layerInfos: [{  
      layer: featurelayer,  
      title: "Name"  
    }]  
  });  
  // 3. view の UI にウィジェットを追加  
  view.ui.add(legend, "bottom-left");  
});
```

View UI

- **ウィジェットの位置**

- **add() : 追加**
- **move() : 移動**
- **remove() : 削除**

```
view.ui.add(legend, "bottom-left");  
view.ui.add(searchWidget, "top-right");
```


ステップ7：ウィジェットの追加

