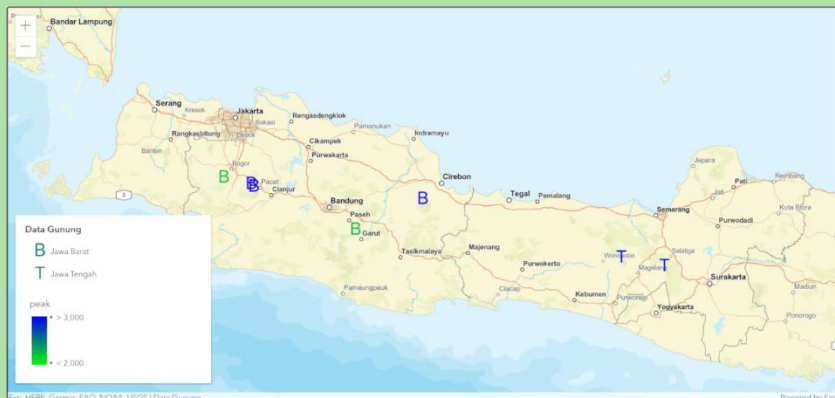
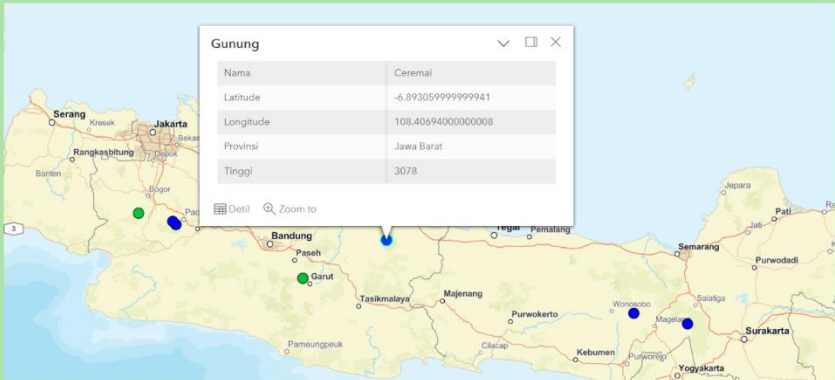


# Contoh Sederhana Penggunaan

# ArcGIS Maps SDK for JavaScript

*Basemap, Renderer, CSVLayer, VisualVariable, Popup (dan action), UniqueValueRenderer, Legend, dan contoh lain.*



**Dr. Noprianto**

**Penerbit PT. Stabil Standar Sinergi**

## Contoh Sederhana Penggunaan ArcGIS Maps SDK for JavaScript

Penulis: Dr. Noprianto

ISBN: 978-602-52770-7-8

Penerbit:

PT. Stabil Standar Sinergi

Alamat	Puri Indah Financial Tower Lantai 6, Unit 0612 Jl. Puri Lingkar Dalam Blok T8, Puri Indah, Kembangan, Jakarta Barat 11610
Website	<a href="http://www.singkong.dev">www.singkong.dev</a>
Email	<a href="mailto:info@singkong.dev">info@singkong.dev</a>

Hak cipta dilindungi undang-undang.

## Daftar Isi

Kata Pengantar .....	2
Persiapan .....	3
Contoh 1: Template kode HTML sederhana.....	5
Registrasi Developer Account dan API Key.....	7
Contoh 2: Menampilkan Basemap .....	11
Contoh 3: Mencari dan Mendapatkan Longitude dan Latitude .....	15
Contoh 4: Menggunakan CSV Layer .....	19
Contoh 5: Menerapkan Renderer pada Layer .....	23
Contoh 6: Menerapkan VisualVariable pada SimpleRenderer .....	25
Contoh 7: Menampilkan Popup.....	27
Contoh 8: Pengaturan Popup Lebih Lanjut.....	31
Contoh 9: Menambahkan Tombol Pada Popup .....	33
Contoh 10: Menangani event click pada MapView.....	39
Contoh 11: Bekerja dengan UniqueValueRenderer .....	43
Contoh 12: Menambahkan Legend .....	45
Daftar Pustaka .....	47

## Kata Pengantar

Buku ini berisi sejumlah contoh source code dan penjelasan langkah demi langkah yang mudah dipahami untuk membuat aplikasi web dengan fitur mapping, menggunakan ArcGIS Maps SDK for JavaScript.

Isi buku telah dirancang untuk pemrogram yang baru mengenal HTML, CSS, dan JavaScript, dan Sistem Informasi Geografi. Walau demikian, beberapa contoh yang disajikan dapat pula diterapkan pada aplikasi web yang ada.

Jakarta, Maret 2023

Dr. Noprianto

**Iklan:** Penulis mengembangkan bahasa pemrograman Singkong dan interpreternya sejak akhir tahun 2019.

Bahasa Singkong merupakan bahasa pemrograman yang case-insensitive (tidak membedakan huruf besar/kecil), dynamically typed (tipe data ditentukan secara dinamis pada saat program berjalan), prosedural, dan interpreted, yang berjalan pada Java Virtual Machine (Java 5.0 atau lebih baru).

Bahasa Singkong dapat digunakan untuk membuat program dengan GUI, terhubung ke sistem database relasional, multithreaded. Dapat bekerja dengan protokol HTTP, dan dapat digunakan untuk membangun aplikasi web (dengan CGI). Selain itu, dapat pula memanggil method Java dan dipanggil dari program Java.

Dengan hanya membutuhkan Java versi 5.0 (atau lebih baru), interpreter bahasa Singkong (dan program yang Anda buat) dapat berjalan pada berbagai sistem operasi, mulai dari versi terbaru, ataupun yang dirilis seperempat abad lalu.

Pada saat buku ini ditulis, telah terdapat 6 buku pemrograman Singkong yang ditulis secara kolaboratif, dan dapat dibaca/didownload gratis dari: <https://singkong.dev>. Beberapa buku lainnya sedang dipersiapkan.

## Persiapan

Apabila Anda telah menyiapkan sebuah komputer, dengan web browser yang masih di-support, terkoneksi ke internet, memiliki alamat email, bersedia menulis setidaknya 21 baris kode HTML/CSS/JavaScript, maka Anda dapat melanjutkan ke bab berikutnya :) Tidak ada software tertentu yang harus diinstal.

---

Berikut adalah perinciannya.

1. Siapkanlah sebuah komputer, yang dilengkapi layar, keyboard, dan mouse/trackpad.
2. Kita akan membutuhkan web browser. Gunakanlah yang masih di-support. Apabila memungkinkan, perbaharuilah browser yang digunakan tersebut.
3. Karena kita perlu mengetik kode program, siapkanlah sebuah editor teks. Yang paling mendasar atau bawaan sistem operasi dapat digunakan. Tidak banyak kode yang akan kita ketik.
4. Kita akan membutuhkan akses internet, termasuk pada saat pengembangan aplikasi.
5. Siapkanlah sebuah direktori kerja. Di dalam direktori tersebut, kita akan membuat sejumlah file HTML, satu per bab dalam buku ini. File-file HTML tersebut akan dibuka dengan web browser yang disebutkan sebelumnya.

6. Kita akan membutuhkan akun developer ArcGIS, dimana pendaftaran perlu dilakukan. Pendaftaran akun akan dibahas di bab tersendiri. Pada saat buku ini ditulis:
  - a. Kita hanya membutuhkan alamat email saja untuk pendaftaran tersebut.
  - b. Apa yang kita bahas dalam buku ini cukup dalam batasan versi free dari akun developer.
  
7. Kita akan menulis sedikit kode HTML mendasar dan beberapa baris CSS, dengan total 21 baris. Selengkapnya, kita akan menulis sejumlah kode JavaScript. Tapi, tidak ada software tambahan apapun yang perlu diinstal.

## Contoh 1: Template kode HTML sederhana

Template yang akan kita buat dan disimpan sebagai `template.html`, akan digunakan sebagai dasar untuk setiap bab dalam buku ini. Berikut ini adalah 21 baris isinya (beberapa baris, karena tidak cukup tempat, isinya akan tampak turun ke baris berikutnya):

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1">
    <title>Hello, world</title>
    <style>
      html, body, #viewDiv {
        padding: 0;
        margin: 0;
        height: 100%;
        width: 100%;
      }
    </style>
    <link rel="stylesheet"
href="https://js.arcgis.com/4.26/esri/themes/lig
ht/main.css">
    <script
src="https://js.arcgis.com/4.26/"></script>
  </head>
  <body>
    <div id="viewDiv"></div>
  </body>
</html>
```

Kode HTML tersebut menyediakan sebuah `div` berukuran penuh ketika ditampilkan di web browser. Bisa kita lihat, kita siapkan sebuah `div`, dengan `id` adalah `viewDiv`, yang nantinya akan menampilkan sebuah map.

Karena kita menggunakan ArcGIS Maps SDK for JavaScript, maka kita perlu menggunakan CSS dan JavaScript dari ArcGIS, sebagaimana `<link>` dan `<script>` pada kode tersebut.

Lalu, dimanakah nantinya kita akan menuliskan kode JavaScript yang dibahas dalam buku ini? Tepat setelah `</script>` dan sebelum `</head>`.

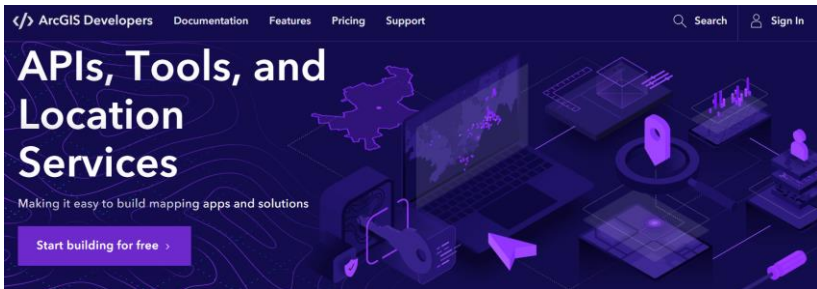
Dengan terkoneksi ke internet (untuk CSS dan JavaScript dari ArcGIS), bukalah file `template.html` tersebut dengan web browser. Tentunya akan berupa halaman kosong, dengan title Hello, World.



## Registrasi Developer Account dan API Key

Untuk contoh-contoh berikutnya dalam buku ini, kita akan membutuhkan API Key. Sebelumnya, kita perlu mendaftar developer account.

Pada saat buku ini ditulis, pendaftaran bisa dimulai dengan mengunjungi <https://developers.arcgis.com>.



Setelah itu, kliklah tombol Start building for free.

Pada form yang tampil berikutnya, kita dapat memilih untuk mendaftar dengan akun yang mungkin dimiliki pada sejumlah platform lain, atau dengan mengisi form pendaftaran.

Apabila memilih untuk mengisi form secara manual, kita akan diminta untuk mengisi:


- First name dan last name
- Email
- Username
- Password dan konfirmasinya
- Security question dan security answer

Setelah itu, pastikanlah untuk menyetujui license agreement dan privacy policy, dan klik tombol Create developer account.

First name


Last name

Email


 You will be required to confirm your email after signing up.

Organization name (optional)

Username

 6 to 24 characters. Letters, numbers and '@\_-' only.

Password

 At least 8 characters. At least 1 letter and 1 number. Cannot match username.

Confirm password

Security question

Security answer

I accept and agree to be legally bound by the [Esri Master License Agreement](#) and [Privacy Policy](#). You can review the [Terms of Use and Privacy Policy](#) in other select languages.

Create developer account

Setelah itu, dalam hanya beberapa menit, proses registrasi pun dapat dilanjutkan. Kita akan dibawa ke dashboard developer, dimana, pada waktu buku ini ditulis, sebelumnya akan diminta untuk mengupdate preferensi. Dalam hal ini, penulis memilih:

- ArcGIS Maps SDK for JavaScript

- I want to build mapping applications using ArcGIS location services.

Kemudian klik Update dan kembali ke dashboard (menutup dialog preferensi yang tampil).

Konfirmasi alamat email diperlukan. Oleh karena itu, bukalah email yang dikirimkan dan konfirmasilah alamat email yang digunakan. Setelah ini, pendaftaran pun sepenuhnya selesai.

Untuk mendapatkan API key, sebuah String, kita bisa melakukan salah satu dari berikut ketika berada di dashboard:

- Pada bagian Recent API Keys, pada Default API Key, kliklah tombol Click to copy, atau Show value dan kopikan ke editor teks.
- Klik link API Keys atau tombol Manage all API Keys, dimana kita bisa menggunakan Default API Key tadi, atau membuat API Key baru dengan tombol New API Key. Kopikanlah ke editor teks.

Simpanlah API Key tersebut. Kita dapat logout dari dashboard developer.

Bagaimanakah kita akan menggunakan API Key tersebut? Kopikanlah template.html ke template\_api.html, dan editlah file yang baru dikopi dengan menambahkan delapan baris kode berikut setelah `</script>` dan sebelum `</head>`.

```
<script>
  require(
    ["esri/config", "esri/Map", "esri/views/MapView"],
    function (esriConfig, Map, MapView) {
      esriConfig.apiKey = "API KEY";
    }
  );
</script>
```

Sesuaikanlah "API KEY" dengan API Key yang didapatkan sebelumnya.  
Pastikanlah kode tersebut diantara:

```
<script src="https://js.arcgis.com/4.26/"></script>
```

dan

```
</head>
```

Catatan:

- Saat ini, kita hanya menggunakan "esri/config" (sebagai esriConfig).
- Kita belum menggunakan modul "esri/Map" (sebagai Map) dan "esri/views/MapView" (sebagai MapView). Namun, untuk contoh-contoh ke depan, kita akan gunakan keduanya. Oleh karenanya, dituliskan di sini.

## Contoh 2: Menampilkan Basemap

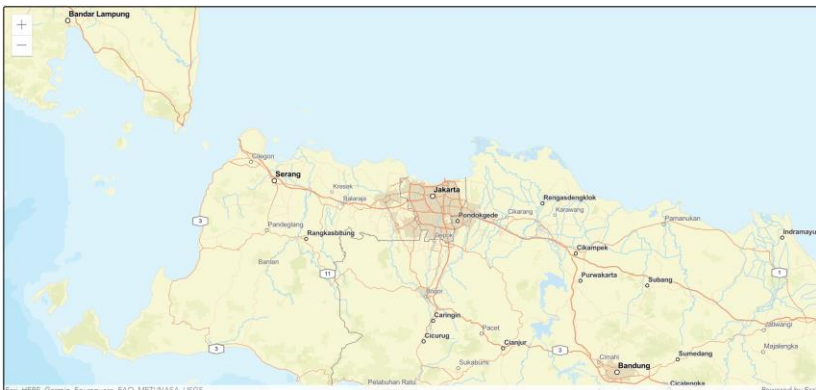
Di contoh ini, kita akan menampilkan sebuah basemap, menggunakan modul "esri/Map" (sebagai Map) dan "esri/views/MapView" (sebagai MapView), melanjutkan contoh sebelumnya.

Kopikanlah `template_api.html` ke `basemap.html`, kemudian tambahkanlah 13 baris kode berikut setelah baris `esriConfig.apiKey`.

```
const map = new Map({
  basemap: "arcgis-streets"
});

const view = new MapView({
  map: map,
  center: [106.73527502302792, -6.1883349955027755],
  zoom: 8,
  container: "viewDiv",
  constraints: {
    snapToZoom: false
  }
});
```

Pastikanlah API Key telah sesuai dan telah terkoneksi ke internet. Kemudian bukalah `basemap.html` tersebut dengan web browser.



Kita dapat melihat tampilan seperti gambar sebelumnya. Sebuah Map, dengan basemap adalah `arcgis-streets`, ditampilkan pada sebuah MapView dengan titik tengah dan level zoom tertentu. Kita bisa scroll dan zoom lokasi tertentu apabila diinginkan.

Sampai di sini, kita mendapatkan hasil yang luar biasa, dengan begitu sedikit jumlah baris kode. Mari kita bahas beberapa hal yang mungkin menarik:

1. Saat ini, map kita tampil penuh di browser. Pada aplikasi Anda, ada kemungkinan map tampil bersama komponen user interface lain. Apabila diperlukan, sesuaikanlah style untuk id `viewDiv` saja di CSS.
2. Kata kunci `const` diperkenalkan pada ECMAScript 2015 (ES2015), yang dapat digunakan untuk membuat konstanta dalam block scoped.
3. Kita menggunakan class `Map` (dari modul `esri/Map`) diantaranya untuk overlay layer untuk ditampilkan secara 2D (MapView) ataupun 3D. Dalam contoh ini, kita menentukan basemap dengan nilai `arcgis-streets`. Apa saja nilai yang bisa ditentukan dengan API Key? Berikut adalah informasi yang penulis dapatkan dari dokumentasi modul. Cobalah untuk menggunakan beberapa basemap yang berbeda.

<code>arcgis-imagery</code>	<code>arcgis-imagery-standard</code>	<code>arcgis-imagery-labels</code>
<code>arcgis-light-gray</code>	<code>arcgis-dark-gray</code>	<code>arcgis-navigation</code>
<code>arcgis-navigation-night</code>	<code>arcgis-streets</code>	<code>arcgis-streets-night</code>
<code>arcgis-streets-relief</code>	<code>arcgis-topographic</code>	<code>arcgis-oceans</code>
<code>arcgis-human-geography</code>	<code>arcgis-human-geography-dark</code>	<code>osm-standard</code>

osm-standard-relief	osm-streets	osm-streets-relief
osm-light-gray	osm-dark-gray	arcgis-terrain
arcgis-community	arcgis-charted-territory	arcgis-colored-pencil
arcgis-nova	arcgis-modern-antique	arcgis-midcentury
arcgis-newspaper	arcgis-hillshade-light	arcgis-hillshade-dark
Info: <a href="https://developers.arcgis.com/javascript/latest/api-reference/esri-Map.html">https://developers.arcgis.com/javascript/latest/api-reference/esri-Map.html</a>		

4. Kita menggunakan class MapView (dari modul esri/views/MapView) untuk menampilkan view 2D dari Map. Perhatikanlah bahwa kita menentukan properti map yang merujuk ke map yang dibuat sebelumnya. Informasi selengkapnya untuk MapView dapat dibaca di: <https://developers.arcgis.com/javascript/latest/api-reference/esri-views-MapView.html>
5. Level of detail (LOD) pada center dapat ditentukan dengan properti zoom, dimana 0 adalah global, 23 adalah sangat detil, dan -1 adalah tanpa LOD.
6. Kita menentukan container, yang nilainya adalah elemen DOM (Document Object Model, yang merepresentasikan dokumen sebagai node dan object). Kita memiliki sebuah div dengan id "viewDiv", dan inilah nilai yang kita gunakan untuk container.
7. Properti center, sebuah Point, merepresentasikan titik tengah, dengan nilai adalah pasangan longitude dan latitude. Perubahan pada center akan langsung mengubah view.

Sebelum mengakhiri contoh ini, Anda mungkin bertanya, dimana persisnya pasangan longitude dan latitude dalam contoh, yaitu [106.73527502302792, -6.1883349955027755]? Lokasi tersebut adalah di Jalan Puri Lingkar Dalam, Kembangan, DKI Jakarta, 11610, lokasi terdekat kantor Singkong.dev yang dapat penulis cari.

Cari? Iya. Kita pun akan membahas contohnya di buku ini, bagaimana membangun aplikasi sederhana yang dapat mencari dan menginformasi longitude dan latitude dari hasil pencarian.



## Contoh 3: Mencari dan Mendapatkan Longitude dan Latitude

Di contoh sebelumnya, penulis ingin mendapatkan latitude dan longitude dari alamat terdekat kantor Singkong.dev. Cara cepat, mengingat kita telah mendaftar developer account ArcGIS, adalah dengan menggunakan layanan geocoding. Geocoding adalah proses konversi dari alamat ke posisi pada permukaan bumi.

Lebih lanjut lagi, kita mungkin ingin menyediakan cara pencarian yang mudah. Misal dengan sebuah text box yang dilengkapi autocomplete. Dengan ArcGIS SDK for JavaScript, kita hanya membutuhkan beberapa baris kode saja. Tepatnya, menambah delapan baris kode dari contoh sebelumnya.

Kopikanlah basemap.html ke search.html. Kemudian, kita tambah modul yang digunakan. Sesuaikanlah statement require di contoh sebelumnya dari:

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

Menjadi:

```
require([
  "esri/config", "esri/Map",
  "esri/views/MapView", "esri/widgets/Search"],
  function (esriConfig, Map, MapView, Search) {
```

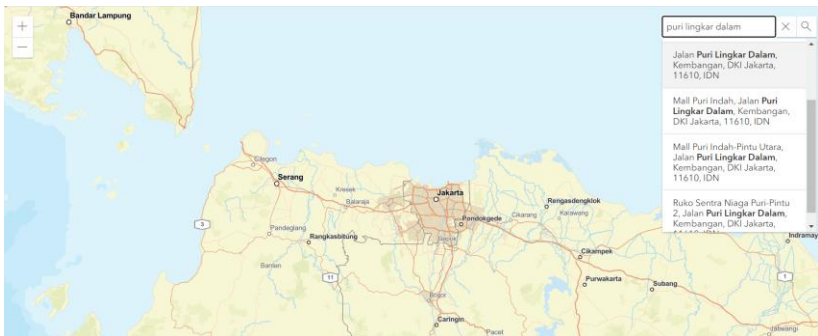
Dengan demikian, kita menggunakan widget Search, dari modul esri/widgets/Search sebagai text box pencarian. Dan, berikut adalah delapan baris yang perlu kita tambahkan:

```

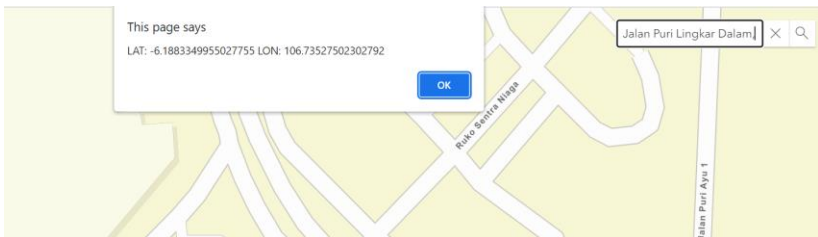
const search = new Search({
  view: view
});
view.ui.add(search, 'top-right');
search.on('select-result', function(e) {
  const g = e.result.feature.geometry;
  alert('LAT: ' + g.latitude + ' LON: ' +
g.longitude);
});

```

Dengan kondisi terkoneksi ke internet, bukalah search.html. Tampilannya mirip dengan contoh sebelumnya, namun terdapat sebuah text box pencarian di sisi kanan atas.



Pada gambar sebelumnya, penulis mengetik: puri lingkaran dalam, dan beberapa hasil pencarian yang cocok akan ditampilkan. Penulis kemudian klik pada baris yang di-highlight.



Dengan cara demikian, kita mendapatkan latitude dan longitude yang digunakan pada contoh sebelumnya.

Sekarang, mari kita bahas beberapa hal berikut:

1. Pertama, kita membuat widget Search dan melewati view ke MapView:

```
const search = new Search({  
  view: view  
});
```

Nantinya, kita akan tambahkan ke MapView lewat properti ui.

2. Kemudian, kita menambahkan widget Search ke MapView di baris berikut:

```
view.ui.add(search, 'top-right');
```

Ini merupakan `esri/views/ui/DefaultUI`. Kita kemudian memanggil method `add`, yang diturunkan dari `esri/views/ui/UI`. Method `add` dapat menerima argumen `position`, yang saat ini, kita berikan nilai `top-right`. Nilai-nilai yang diterima adalah:

<code>bottom-leading</code>	<code>bottom-left</code>
<code>bottom-right</code>	<code>bottom-trailing</code>
<code>top-leading</code>	<code>top-left</code>
<code>top-right</code>	<code>top-trailing</code>
<code>manual</code>	

Apabila tidak diberikan, maka posisi default adalah `manual`, yang memungkinkan penggunaan CSS.

3. Widget Search memiliki event, yang dalam contoh ini, adalah `select-result`, yang terjadi ketika hasil pencarian dipilih. Event-event lain dapat dibaca di <https://developers.arcgis.com/javascript/latest/api-reference/esri-widgets-Search.html>

Widget ini memiliki method `on`, yang diturunkan dari `esri/widgets/Widget`. Method `on` digunakan untuk mendaftarkan event handler. Dalam contoh ini, ketika hasil pencarian dipilih, kita menyediakan sebuah fungsi, yang dibahas berikut.

4. Ketika fungsi tersebut dipanggil, kita menggunakan argumen yang dilewatkan untuk mendapatkan informasi latitude dan longitude. Berdasarkan dokumentasi event `select-result`, kita mengetahui bahwa properti `result.feature` adalah sebuah `esri/Graphic`. Salah satu properti dari `Graphic` adalah `geometry` (`esri/geometry/Geometry`). Properti `declaredClass` dalam hal ini adalah `esri.geometry.Point` (berdasarkan informasi dari developer tools di browser). Pada `esri/geometry/Point`, yang merupakan subclass dari `Geometry`, tersedia properti latitude dan longitude.

Seru bukan? Berikutnya, kita akan menambahkan simbol-simbol pada map, berdasarkan informasi latitude dan longitude. Penulis akan menggunakan beberapa contoh lokasi, namun Anda mungkin mencari dan menambahkan sendiri. Ini adalah salah satu alasan contoh ini dibahas cukup awal.

## Contoh 4: Menggunakan CSV Layer

Di contoh menampilkan basemap sebelumnya, kita bisa melakukan scroll dan zoom lokasi tertentu. Untuk sekedar melihat map, tentunya fasilitas yang disediakan sangatlah berguna.

Akan tetapi, kita mungkin perlu memberikan penanda tertentu pada map. Sebagai contoh, penulis ingin menandai beberapa lokasi dengan simbol tertentu, dimana, untuk setiap simbol, kita bisa terapkan aksi tertentu. Seperti dicontohkan pada cover depan buku.

Untuk kebutuhan tersebut, kita dapat menggunakan layer-layer, yang dapat ditambahkan ke (atau dihapus dari) map. Terdapat sejumlah jenis layer yang dapat kita gunakan, dengan fungsi dan karakteristik yang berbeda. Salah satunya adalah CSVLayer, yang merupakan layer point.

Sebagaimana namanya, layer ini berbasiskan pada format CSV. CSV merupakan singkatan dari Comma-Separated Values dan merupakan format file teks yang merepresentasikan data tabular. Setiap baris data (atau record) dapat terdiri dari satu atau lebih field, dipisahkan karakter tertentu, atau koma (sebagaimana nama format filenya).

CSVLayer dapat dibuat dengan mengatur properti url. Properti ini merupakan sebuah String yang dapat merujuk pada:

- Sumber data di server. Apabila berada pada domain berbeda, maka server perlu mengaktifkan Cross-Origin Resource Sharing, pengaturan HTTP header yang mengijinkan sumber daya diakses dari origin berbeda.
- Blob untuk data di memori. Kita tuliskan langsung isi file CSV nya di kode program. Ini yang akan kita gunakan dalam buku ini.

Sebagai contoh sederhana, mari kita contohkan beberapa baris berikut, yang merupakan data beberapa gunung di Indonesia. Nilai latitude dan longitude tentunya didapatkan dengan contoh aplikasi pencarian yang telah kita bahas sebelumnya.

id	name	latitude	longitude	province	peak
1	Ceremai	-6.893059999999941	108.40694000000008	Jawa Barat	3078
2	Gede	-6.790249999999955	106.98452000000009	Jawa Barat	3008
3	Pangrango	-6.769999999999922	106.96361000000007	Jawa Barat	3019
4	Salak	-6.715559999999939	106.73389000000007	Jawa Barat	2211
5	Guntur	-7.148419999999949	107.83991000000009	Jawa Barat	2249
6	Merbabu	-7.453609999999969	110.43944000000005	Jawa Tengah	3145
7	Sumbing	-7.381939999999922	110.07528000000009	Jawa Tengah	3371

Catatan pencarian yang dilakukan:

Gunung	Kata kunci
Ceremai	Mount Cereme, Jawa Barat, IDN
Gede	Mount Gede, Jawa Barat, IDN
Pangrango	Gunung Pangrango, Jawa Barat, IDN
Salak	Mount Salak, Jawa Barat, IDN
Guntur	Mount Guntur, Jawa Barat, IDN
Merbabu	Mount Merbabu, Jawa Tengah, IDN
Sumbing	Mount Sumbing, Jawa Tengah, IDN

Kopikanlah basemap.html ke csvlayer.html, kemudian editlah sebagaimana pembahasan berikut.

Berikut adalah representasi data tersebut dalam kode JavaScript, menggunakan kutip backtick (`) dan dituliskan per baris untuk setiap baris data gunung:

```
const csv = `id|name|latitude|longitude|province|peak
1|Ceremai|-6.893059999999941|108.40694000000008|Jawa
Barat|3078
2|Gede|-6.790249999999955|106.98452000000009|Jawa Barat|3008
3|Pangrango|-6.769999999999922|106.96361000000007|Jawa
Barat|3019
4|Salak|-6.715559999999939|106.73389000000007|Jawa Barat|2211
5|Guntur|-7.148419999999949|107.83991000000009|Jawa
Barat|2249
6|Merbabu|-7.453609999999969|110.43944000000005|Jawa
Tengah|3145
7|Sumbing|-7.381939999999922|110.07528000000009|Jawa
Tengah|3371`;
```

Setelah itu, kita buat sebuah blob:

```
const blob = new Blob([csv], {
  type: 'text/csv'
});
```

Dan, pada akhirnya, membuat sebuah CSVLayer:

```
const layer = new CSVLayer({
  url: URL.createObjectURL(blob),
  copyright: 'Data Gunung'
});
```

Dan, kita tambahkan layer ke map:

```
map.add(layer);
```

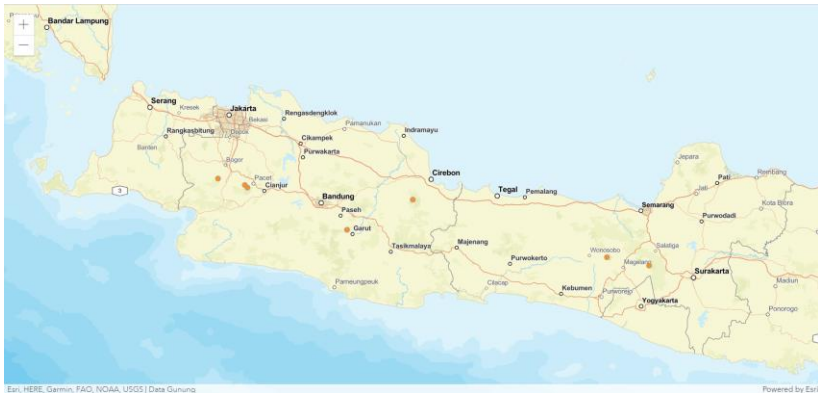
Perhatikanlah, sebagaimana dalam contoh pencarian, kita menggunakan modul tambahan, sehingga sesuaikanlah statement require menjadi:

```
require(
  ["esri/config", "esri/Map",
  "esri/views/MapView", "esri/layers/CSVLayer"],
  function (esriConfig, Map, MapView, CSVLayer) {
```

Kemudian, kita ubah center dan zoom pada view menjadi:

```
center: [108.40694000000008, -6.893059999999941],  
zoom: 7,
```

Dengan terkoneksi ke internet, bukalah file csvlayer.html. Kita bisa melihat tujuh simbol telah ditambahkan ke map, beserta keterangan Data Gunung di bawah map. Berikut adalah contoh gambarnya:



Walau mungkin tidak terlalu terlihat jelas pada contoh gambar tersebut, kita telah berhasil memvisualisasikan data tertentu pada map, dengan sebuah layer.

Di contoh berikut, kita akan mengubah simbol pada map agar terlihat lebih jelas.



## Contoh 5: Menerapkan Renderer pada Layer

Di contoh ini, kita akan mengubah simbol pada contoh sebelumnya menjadi:

- Circle, seperti sebelumnya, tapi lebih besar dengan warna hijau.
- Kotak dengan warna biru.

Untuk kebutuhan tersebut, kita akan menerapkan renderer pada layer yang kita buat sebelumnya. Kopikanlah terlebih dahulu file csvlayer.html ke renderer.html. Kemudian, editlah file renderer.html.

Pertama, kita membuat renderernya terlebih dahulu, dimana diantaranya, kita bisa mengubah simbol. Kita akan menggunakan class SimpleRenderer, yang di autocast (sehingga tidak perlu diimport secara eksplisit).

Sebelum baris const layer, tambahkanlah kode berikut:

```
const renderer = {
  type: 'simple',
  symbol: {
    type: 'simple-marker',
    size: 10,
    color: [0, 255, 0]
  }
};
```

Kemudian, tambahkanlah renderer tersebut ketika membuat CSVLayer, sehingga pembuatannya menjadi:

```
const layer = new CSVLayer({
  url: URL.createObjectURL(blob),
  renderer: renderer,
  copyright: 'Data Gunung'
});
```

Dengan terkoneksi ke internet, bukanlah file renderer.html. Lihatlah bahwa simbol yang tampil mungkin dapat terlihat lebih jelas, dengan warna dan ukuran berbeda:



Dalam contoh tersebut, selain ke SimpleRenderer, juga terdapat autocast ke SimpleMarkerSymbol. Untuk mengubah ke kotak warna biru, kita cukup menambahkan style berupa square dan mengubah warnanya, seperti contoh berikut (pada renderer):

```
symbol: {  
  type: 'simple-marker',  
  style: 'square',  
  size: 10,  
  color: [0, 0, 255]  
}
```



Style bawaannya dapat berupa: circle (default), square, cross, x, diamond, triangle, dan path. Kita akan membahas contoh lebih lanjutnya di bab lain.

## Contoh 6: Menerapkan VisualVariable pada SimpleRenderer

Di contoh sebelumnya, semua gunung ditampilkan dengan warna simbol yang sama. Sekarang, kita ingin menampilkan warna yang berbeda untuk kategori ketinggian yang berbeda. Dalam data, kita memiliki kolom peak, dimana terdapat gunung dengan ketinggian antara 2000 sampai 3000 dan lebih dari 3000.

Untuk kebutuhan tersebut, kita bisa gunakan visual variable pada renderer yang kita buat sebelumnya. Pada renderer, kita gunakan properti `visualVariables`, yang merupakan autocast ke `esri/renderers/visualVariables/VisualVariable`. Nilainya dapat berupa pengaturan tipe, field, dan stops.

Kita ingin mengubah warna, maka tipenya adalah `color`. Kemudian, dasar pengubahan kita adalah kolom peak, maka field adalah `peak`. Apabila kategori ketinggian 2000 ingin diberi warna hijau, dan ketinggian 3000 ingin diberi warna biru, maka stops diberikan dengan value dan color tersebut.

Kopikanlah `renderer.html` ke `visual.html` dan editlah. Berikut adalah kode untuk renderer (style symbol default, tanpa menentukan color):

```
const renderer = {
  type: 'simple',
  symbol: {
    type: 'simple-marker',
    size: 10,
  },
  visualVariables: [{
    type: 'color',
    field: 'peak',
    stops: [
      {value: 2000, color: '#00FF00'},
      {value: 3000, color: '#0000FF'}
    ]
  }
];
```

Dengan terkoneksi ke internet, bukanlah file visual.html. Kini, kita bisa melihat penggunaan warna yang berbeda untuk simbol, berdasarkan kategori ketinggian tertentu. Luar biasa, bukan?



Di bab berikutnya, kita akan melihat contoh yang dapat kita lakukan ketika klik dilakukan pada simbol.

## Contoh 7: Menampilkan Popup

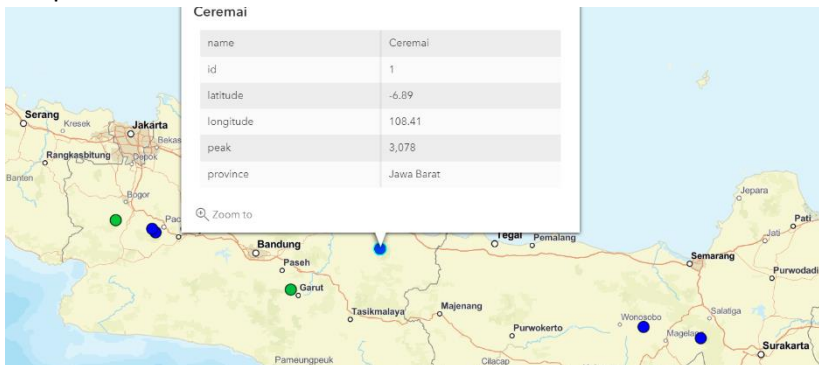
Bayangkanlah seperti ini. Untuk setiap simbol pada gunung, ketika di klik, sebuah popup akan tampil, lengkap dengan kolom-kolom yang telah kita tentukan sebelumnya.

Di bab ini, kita akan mulai dengan cara yang benar-benar mudah dan cara mudah. Di bab lain, kita akan membahas cara yang lebih sulit.

Cara yang benar-benar mudah dapat dilakukan dengan menambah satu baris kode. Kopikanlah file `visual.html` ke `popup_simple.html`, dan tambahkanlah baris berikut:

```
view.popup.defaultPopupTemplateEnabled = true;
```

Dengan terkoneksi ke internet, bukalah file `popup_simple.html`. Kliklah salah satu gunung dan popup seperti contoh berikut akan tampil.



Bagaimana? Benar-benar mudah bukan? Dan harusnya tidak membutuhkan pembahasan tersendiri. Jadi, bab ini selesai di sini? Sayangnya, belum.

Walau mungkin contoh sebelumnya dapat memenuhi kebutuhan, kita mungkin ingin menampilkan field tertentu saja, dengan label berbeda. Atau dengan judul berbeda.

Kita akan gunakan cara yang mudah. Kopikanlah file visual.html ke popup.html dan editlah.

Cara mudah akan dimulai dengan penggunaan properti popupTemplate pada CSVLayer. Ini akan di-autocast ke esri/PopupTemplate. Sebenarnya, properti penentu lain adalah popupEnabled. Tapi, karena defaultnya adalah true, maka kita fokus pada popupTemplate tersebut.

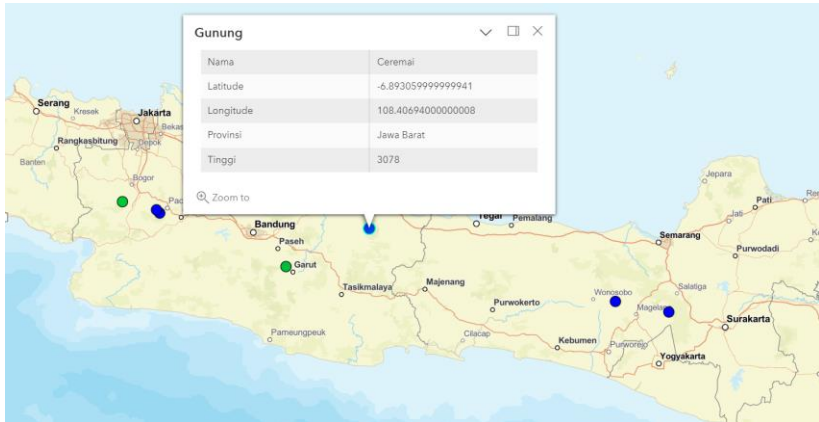
Berikut ini adalah contoh templatennya, yang ditempatkan sebelum const layer:

```
const template = {
  title: 'Gunung',
  outFields: ['*'],
  content: [{
    type: 'fields',
    fieldInfos: [
      {fieldName: 'name', label: 'Nama'},
      {fieldName: 'latitude', label: 'Latitude'},
      {fieldName: 'longitude', label: 'Longitude'},
      {fieldName: 'province', label: 'Provinsi'},
      {fieldName: 'peak', label: 'Tinggi'},
    ]
  }]
};
```

Dan, ubahlah pembuatan layer menjadi berikut, dengan menambahkan popupTemplate:

```
const layer = new CSVLayer({
  url: URL.createObjectURL(blob),
  renderer: renderer,
  popupTemplate: template,
  copyright: 'Data Gunung'
});
```

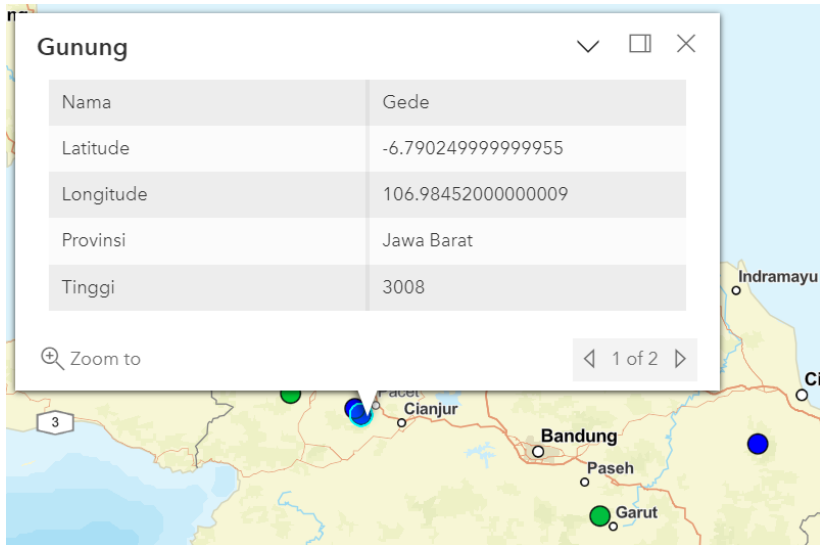
Dengan terkoneksi ke internet, bukanlah file popup.html. Kliklah pada salah satu gunung, dan popup sesuai template akan ditampilkan, seperti pada gambar berikut.



Bisa dilihat, kita memberikan label untuk setiap kolom. Tentunya, per template tersebut, kita juga bisa tidak menampilkan semua kolom.

Template datang dengan bonus misal:

- Di samping tombol close, terdapat tombol dock. Apabila diklik, dengan popup di posisi dock, untuk setiap gunung yang diklik, posisi popup akan tetap. Berbeda ketika tidak di-dock, posisi popup akan mengikuti lokasi.
- Selain itu, terdapat tombol Zoom to, yang apabila diklik, akan zoom ke lokasi. Tombol ini dapat diklik berkali-kali dalam contoh kita.
- Lebih menarik lagi, cobalah untuk klik pada simbol gunung Pangrango atau gunung Gede, yang saling bersebelahan. Perhatikanlah di baris yang sama dengan tombol Zoom to, akan ditampilkan 1 of 2 yang bisa dinavigasikan, seperti pada gambar berikut.



Bagaimana kalau kita ingin sedikit mengubah bonus yang disertakan tersebut? Misal dengan dock secara default. Atau dengan tidak menampilkan navigas 1 of 2 tersebut. Mari kita bahas di bab berikut.



## Contoh 8: Pengaturan Popup Lebih Lanjut

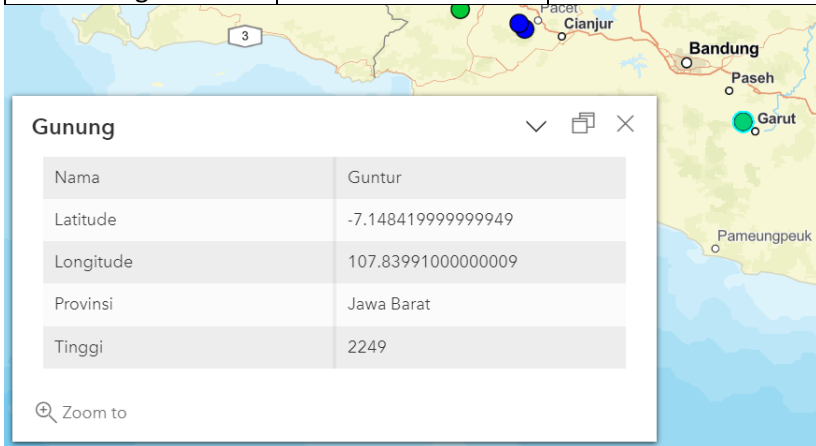
Bab ini akan singkat dan melibatkan hanya beberapa baris kode. Mari kita lihat contoh pengaturan pertama, dimana kita akan dock popup secara default.

Kopikanlah popup.html ke popup\_dock.html, dan editlah dengan menambah beberapa baris berikut di akhir:

```
view.popup = {  
  dockEnabled: true,  
  dockOptions: {  
    breakpoint: false,  
    position: 'bottom-left'  
  }  
};
```

Dengan terkoneksi ke internet, bukalah file popup\_dock.html. Ketika klik pada gunung dilakukan, bisa dilihat bahwa posisi popup akan selalu di sisi kiri bawah. Berikut adalah nilai yang mungkin untuk posisi (default adalah auto):

auto	top-left	top-center
top-right	bottom-left	bottom-center
bottom-right		



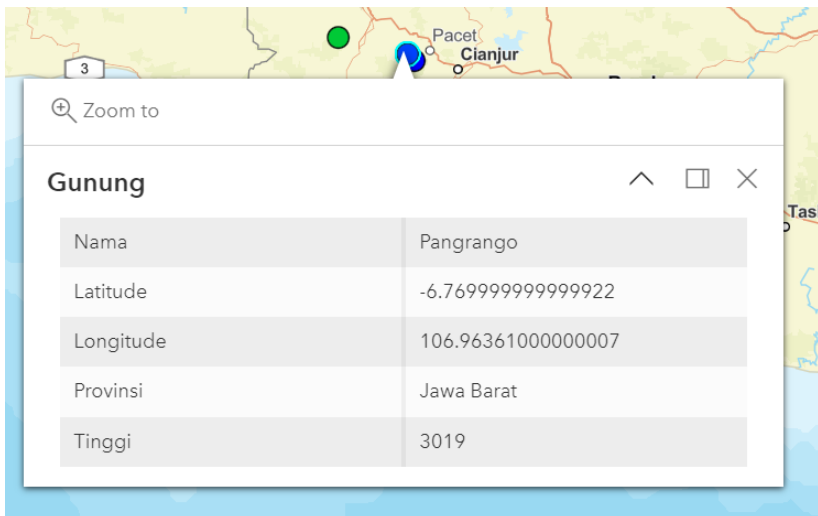
Pengaturan berikutnya adalah terkait navigasi fitur seperti 1 of 2 yang dibahas di bab sebelumnya. Untuk tidak menampilkan fitur ini, kita hanya perlu mengubah satu baris kode.

Kopikanlah popup.html ke popup\_nav.html, dan editlah dengan menambah baris berikut:

```
view.popup.visibleElements.featureNavigation = false;
```

**Catatan:** kode tersebut berhasil pada ArcGIS Maps SDK for JavaScript versi 4.25. Namun di konfigurasi sistem yang penulis gunakan, untuk versi 4.26, pada saat buku ini ditulis, penulis tidak mengetahui kenapa tidak berfungsi sama.

Dengan terkoneksi ke internet, bukanlah file popup\_nav.html tersebut. Bisa kita lihat, ketika klik pada Gunung Gede dan Gunung Pangrango, tidak lagi ditampilkan navigasi 1 of 2 seperti sebelumnya.



## Contoh 9: Menambahkan Tombol Pada Popup

Ketika simbol pada suatu gunung di klik, kita ingin menampilkan cuaca atau prakiraannya pada kisaran waktu tertentu. Kita tidak memiliki data ini pada CSV yang digunakan. Kita bahkan tidak memiliki data ini di server dan perlu memintanya dari pihak lain.

Dengan demikian, pada saat popup tampil, kita akan siapkan satu tombol, yang apabila di klik, dapat mengirim permintaan dan menampilkan hasilnya. Apabila hasilnya berupa data tabular, kita tampilkan dalam sebuah tabel.

Agar pembahasan dalam buku ini tetap sederhana, bagian meminta ke pihak lain tidak dibahas secara langsung dan data tabular akan disimulasikan dengan data dummy.

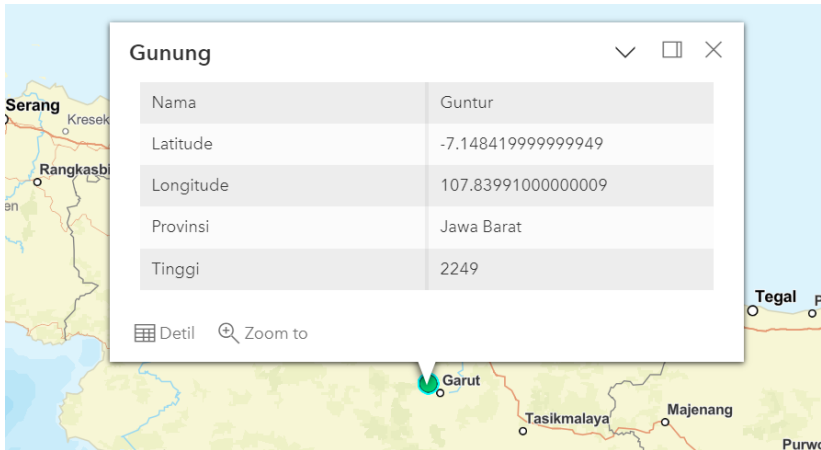
Kopikanlah `popup.html` ke `popup_button.html` dan editlah dengan menambahkan beberapa baris berikut sebelum `const template`:

```
const actionDetail = {  
  id: 'btnActionDetail',  
  title: 'Detil',  
  className: 'esri-icon-table'  
};
```

Kemudian, tambahkanlah actions pada template, misal setelah `title`:

```
actions: [actionDetail],
```

Dengan terkoneksi ke internet, bukalah file `popup_button.html` tersebut. Cobalah untuk klik pada salah satu gunung. Berdekatan dengan tombol Zoom to, sebuah tombol dengan label Detil (dengan icon tabel) akan tampil. Sayangnya, tidak ada yang terjadi begitu tombol tersebut di klik. Kita memang belum menulis kodenya.



Sebelum menangani klik, mari kita bahas beberapa hal yang mungkin menarik.

1. Salah satu properti dari PopupTemplate adalah actions. Itulah yang kita tambahkan pada definisi template.
2. Properti action tersebut menerima nilai berupa Collection (esri/core/Collection) dari ActionButton (esri/support/actions/ActionButton) ataupun ActionToggle (esri/support/actions/ActionToggle). Dalam hal ini, kita membuat satu ActionButton.
3. Contoh properti untuk Class ActionButton yang kita gunakan adalah:
  - a. id: berguna untuk membedakan dari id ketika menangani event trigger-action. Ini belum kita lakukan, dan oleh karenanya, tidak ada yang terjadi ketika tombol diklik.
  - b. title: label tombolnya.

- c. `className`: barangkali, ini adalah yang paling menarik, dimana kita menentukan icon dari tombol. Properti ini merupakan nama class CSS. Dalam hal ini, kita otomatis dapat menggunakan Esri Icon Font. Untuk nama-nama class yang disediakan, kunjungilah:  
<https://developers.arcgis.com/javascript/latest/esri-icon-font/>

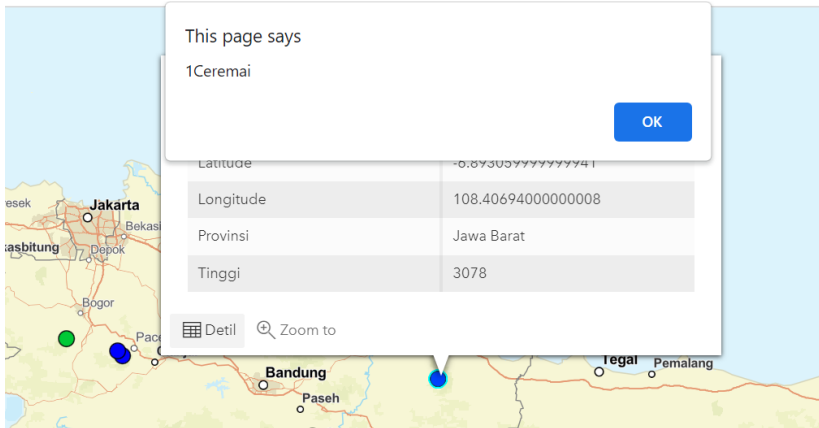
Dengan memahami rincian tersebut, tentunya kita dapat dengan mudah menambahkan tombol lainnya. Cukup tambahkan sebagai elemen untuk actions.

Kini, kita akan menangani klik. Kita tahu bahwa harus menangani event `trigger-action`. Tambahkan beberapa baris berikut di akhir kode JavaScript kita:

```
view.popup.on('trigger-action', function(e) {  
    if (e.action.id == 'btnActionDetail') {  
        let a = view.popup.selectedFeature.attributes;  
        let i = a.id;  
        let n = a.name;  
        alert(i + n);  
    }  
});
```

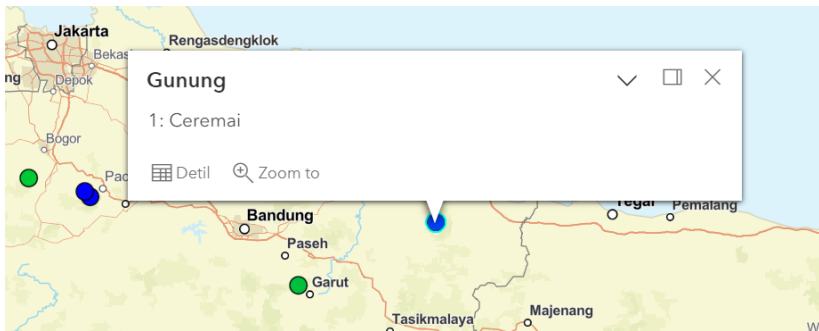
Bukalah kembali `popup_button.html`. Sebagai contoh, penulis akan klik pada simbol di Gunung Ceremai. Ketika tombol Detil di klik, yang terjadi adalah sebuah alert dialog akan ditampilkan, seperti gambar di halaman berikut.

Dalam aplikasi di dunia nyata, data akan memiliki id atau penanda unik tertentu, yang akan dilewatkan dalam permintaan ke server, sebagai contoh. Atribut yang dipilih tersedia sebagaimana variabel `a` dalam kode.



Untuk menampilkan isi dialog, alih-alih dengan alert, kita bisa set `view.popup.content`. Apabila kita mengganti baris `alert` dengan `view.popup.content`, kita akan mendapatkan hasil seperti gambar berikut:

```
view.popup.content = a.id + ': ' + n;
```



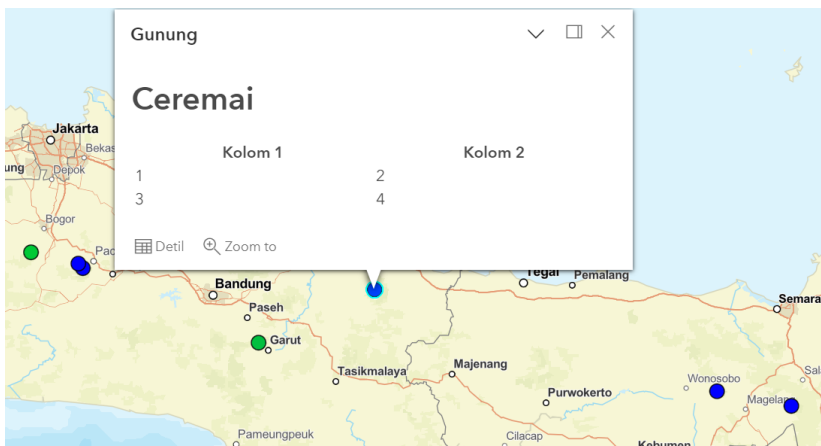
Dengan demikian, untuk mensimulasikan data dalam bentuk tabular, kita cukup memberikan string kode HTML untuk tabel ke `view.popup.content`, seperti contoh berikut:

```

let a = view.popup.selectedFeature.attributes;
let i = a.id;
let n = a.name;
let h =
  <h1>${n}</h1>
  <table>
    <tr>
      <th>kolom 1</th>
      <th>kolom 2</th>
    </tr>
    <tr>
      <td>1</td>
      <td>2</td>
    </tr>
    <tr>
      <td>3</td>
      <td>4</td>
    </tr>
  </table>
;
view.popup.content = h;

```

Perhatikanlah backtick yang digunakan pada kode tersebut. Ketika dijalankan, hasilnya adalah seperti pada gambar berikut.



Kode HTML tersebut tentunya bisa dihasilkan dari data yang diterima dari server. Sebagaimana disebut sebelumnya, kita tidak mencontohkannya dalam buku ini. Walau demikian, pengembang

aplikasi web tentunya bisa menggunakan beberapa opsi seperti jQuery atau Web API yang lebih baru.

Contoh penggunaan jQuery get misalnya:

```
jQuery.get('URL', function(d, s) {  
    ....  
    view.popup.content = ....  
})
```

Catatan:

- Gantilah URL dengan URL yang digunakan.
- Karakter . . . . dalam contoh tersebut dimaksudkan untuk diganti dengan kode yang sesungguhnya.
- Dalam contoh kode tersebut, d adalah data yang dikembalikan. Anda mungkin ingin memroses lebih lanjut, misal untuk data JSON yang didapatkan.



## Contoh 10: Menangani event click pada MapView

Sebelumnya, kita telah membahas cara yang benar-benar mudah dan cara yang mudah untuk menampilkan popup. Kini, tiba saatnya, kita membahas cara yang lebih sulit.

Anda mungkin bertanya. Untuk apa? Tentunya bukan hanya untuk menambah isi dari buku ini. Juga bukan hanya untuk dokumentasi andaikata penulis nantinya lupa akan cara ini. Alasan yang mungkin terpikir adalah:

- Selama ini, kita bekerja dengan fungsi-fungsi yang disediakan dari SDK.
- Untuk menampilkan popup misalnya, kita menggunakan class yang disediakan, yang berfungsi sangat baik.
- Akan tetapi, penulis tidak mengetahui persis cara penggunaan class-class yang ada, misal ketika informasi yang didapatkan dari klik simbol di map, ingin ditampilkan di luar container untuk MapView. Misal di div lain, atau modal dialog lain.

Yang pertama kita tahu adalah bahwa MapView memiliki sejumlah event yang bisa ditangani dengan mendaftarkannya lewat method on. Karena dalam hal ini kita tertarik dengan klik, maka kita akan menangani event click.

Ketika berhasil, sejumlah properti berikut akan dikembalikan:

Properti	Tipe/Nilai
mapPoint	Point
x	Number
y	Number

button	Number
buttons	0   1   2
type	"click"
stopPropagation	Function
timestamp	Number
native	Object

(dokumentasi: <https://developers.arcgis.com/javascript/latest/api-reference/esri-views-MapView.html#event-click>)

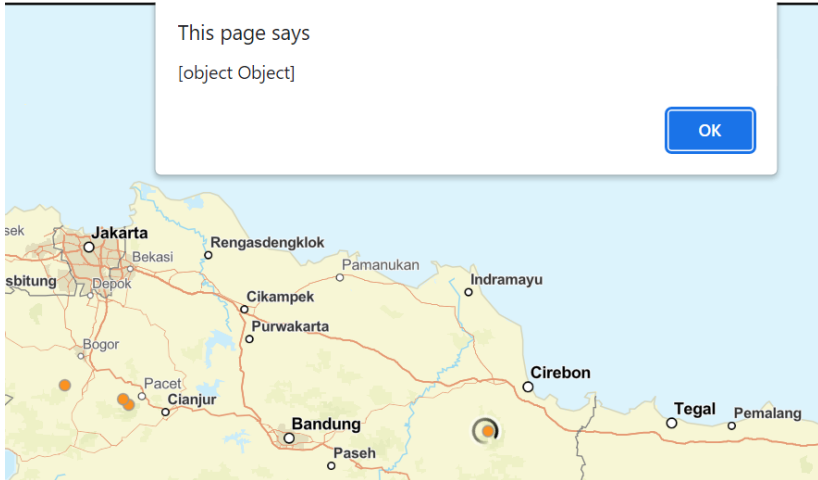
Untuk kebutuhan kita, yang dibutuhkan adalah x dan y.

Kemudian, salah satu method yang tersedia adalah hitTest, yang akan mengembalikan Promise dari hit test result. Method ini membutuhkan argumen berupa ScreenPoint dimana x dan y yang dibutuhkan bisa didapatkan dari penanganan event.

Mari kita bahas dengan contoh berikut. Untuk kebutuhan ini, kopikanlah csvlayer.html ke click.html, dan tambahkanlah beberapa baris berikut:

```
view.on('click', function(e) {
  const p = {x: e.x, y: e.y};
  view.hitTest(p).then(function(r) {
    if (r.results.length) {
      let g = r.results.filter(function(res) {
        return res.graphic.layer == layer;
      });
      if (Array.isArray(g) && g.length) {
        alert(g);
      }
    }
  });
});
```

Dengan terkoneksi ke internet, bukalah click.html. Ketika klik dilakukan pada simbol gunung, kita akan mendapatkan Object, sebagaimana gambar berikut.

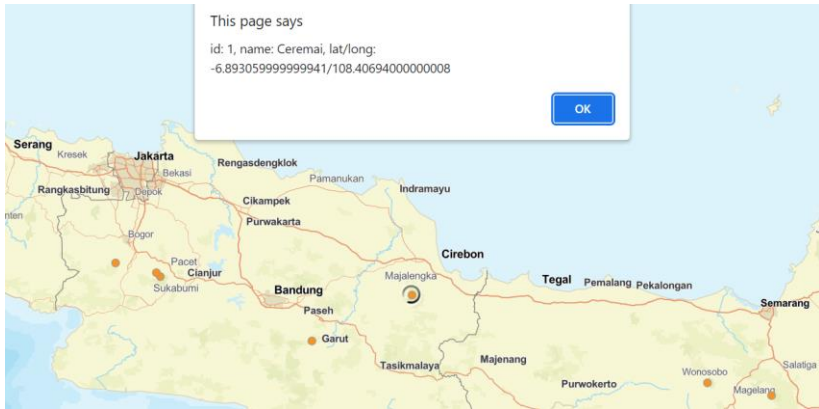


Ini berarti kita sudah mendekati hasilnya. Melalui developer tools, kita dapat melihat bahwa elemen pertama dalam array yang dikembalikan memiliki atribut `graphic`, yang selanjutnya memiliki `attributes`, yang selanjutnya memiliki atribut berupa kolom data CSV.

Mari kita perbaiki kodenya. Perhatikanlah penggunaan backtick:

```
view.on('click', function(e) {
  const p = {x: e.x, y: e.y};
  view.hitTest(p).then(function(r) {
    if (r.results.length) {
      let g = r.results.filter(function(res) {
        return res.graphic.layer == layer;
      });
      if (Array.isArray(g) && g.length) {
        let a = g[0].graphic.attributes;
        alert(`id: ${a.id}, name: ${a.name}, lat/long:
        ${a.latitude}/${a.longitude}`);
      }
    }
  });
});
```

Dan, berikut adalah contoh layarnya:



Dalam bab ini, kita menggunakan alert. Namun, tentu saja, kita bisa menampilkannya dengan cara lain.

## Contoh 11: Bekerja dengan UniqueValueRenderer

Contoh-contoh sebelumnya tidak menggunakan kolom province. Dengan demikian, walaupun kita dapat membedakan gunung mana yang di atas 2000 meter dan yang mana yang di atas 3000 meter, kita tidak bisa membedakan di provinsi mana gunung tersebut berada. Tentunya berdasarkan simbol, dan bukannya dengan mengamati map.

Di bab ini, kita akan menggunakan UniqueValueRenderer, yang memungkinkan simbol dibedakan berdasarkan atribut kategori, yang dalam data CSV kita, adalah kolom province.

Kopikanlah popup\_button.html ke unique.html. Ada beberapa baris yang harus diubah pada renderer kita. Yang pertama adalah kita tidak lagi menggunakan SimpleRenderer. Dengan demikian, type nya perlu diubah menjadi unique-value:

```
type: 'unique-value',
```

Kemudian, kita perlu menentukan field mana yang menjadi dasar, dengan menambahkan properti field:

```
field: 'province',
```

Kita tidak membutuhkan properti symbol dan oleh karenanya, dapat dihapus.

Dan, terakhir, kita perlu menentukan uniqueValueInfos, seperti contoh berikut:

```

uniqueValueInfos: [
  {
    value: 'Jawa Barat',
    symbol: {
      type: 'text',
      text: 'B',
      font: {size: 20},
    },
  },
  {
    value: 'Jawa Tengah',
    symbol: {
      type: 'text',
      text: 'T',
      font: {size: 20},
    },
  },
],

```

Dalam contoh tersebut, untuk setiap nilai unik (provinsi), kita gunakan simbol berbeda. Dalam hal ini, kita menggunakan TextSymbol (esri/symbols/TextSymbol). Simbol-simbol lain tentunya juga dapat digunakan. Sebagai contoh, simple-marker yang kita gunakan sebelumnya.

Berikut adalah contoh layarnya. Dalam hal ini, B mewakili gunung di Jawa Barat dan T mewakili gunung di Jawa Tengah. Visual variable dari tinggi gunung tetap berlaku (warna hijau dan biru).



## Contoh 12: Menambahkan Legend

Ini adalah bab terakhir, dimana kita akan melengkapi map kita dengan sebuah legend. Termasuk penggunaan UniqueValueRenderer dan visual variable. Lebih hebatnya lagi, hanya dengan beberapa baris kode.

Kopikanlah unique.html ke legend.html dan editlah filenya.

Pertama, kita perlu modul esri/widgets/Legend. Dengan demikian, sesuaikanlah statement require menjadi:

```
require([
    "esri/config", "esri/Map",
    "esri/views/MapView", "esri/layers/CSVLayer",
    "esri/widgets/Legend"],
    function (esriConfig, Map, MapView, CSVLayer,
    Legend) {
```

Setelah itu, tambahkan dua baris berikut di akhir kode JavaScript kita:

```
let legend = new Legend({view: view});
view.ui.add(legend, 'bottom-left');
```

Dan, terakhir, kita akan menambahkan title ke CSVLayer, dengan menambah satu baris berikut (sebelum copyright). Per dokumentasi, apabila title tidak diberikan, maka nama service yang akan digunakan.

```
title: 'Data Gunung',
```

Dengan terkoneksi ke internet, bukalah legend.html. Hasilnya adalah seperti pada gambar berikut.





## Daftar Pustaka

Esri. (2023). ArcGIS Maps SDK for JavaScript 4.26.  
<https://developers.arcgis.com/javascript/latest/>

Noprianto. (2020). Mengenal dan Menggunakan Bahasa Pemrograman Singkong. PT. Stabil Standar Sinergi.

Buku ini berisi sejumlah contoh source code dan penjelasan langkah demi langkah yang mudah dipahami untuk membuat aplikasi web dengan fitur mapping, menggunakan ArcGIS Maps SDK for JavaScript.

Isi buku telah dirancang untuk pemrogram yang baru mengenal HTML, CSS, dan JavaScript, dan Sistem Informasi Geografi. Walau demikian, beberapa contoh yang disajikan dapat pula diterapkan pada aplikasi web yang ada.



Dr. Noprianto mengembangkan bahasa pemrograman Singkong dan interpreturnya sejak akhir 2019.

Beliau menyukai pemrograman, dan mendirikan serta mengelola perusahaan pengembangan software dan teknologi Singkong.dev (PT. Stabil Standar Sinergi).

Noprianto menyelesaikan pendidikan doktor ilmu komputer dan telah menulis beberapa buku pemrograman (termasuk Python, Java, dan Singkong).

**singkong.dev**

Alamat: Puri Indah Financial Tower  
Lantai 6, Unit 0612  
Jl. Puri Lingkar Dalam Blok T8  
Kembangan, Jakarta Barat 11610  
Email: [info@singkong.dev](mailto:info@singkong.dev)

ISBN 978-602-52770-7-8



Rp. 50.000