

tt()

Schedule

1. Date and Time in JS
2. Spatial data for the Web
3. Geospatial Apps



Schedule

1. Date and Time in JS
2. Spatial Data for the Web
3. Geospatial Apps



Date Objects: the problem

Different cultures adopt different data formats

Sun calendars / Moon calendars

Political aspects: UTC, summer time / winter time....

Zero state:

- ❖ King's reign (13)
- ❖ Creation (5784)
- ❖ Birth of Jesus (2023)
- ❖ Hidjira (1433)
- ❖

Date Objects: JS's solution

JavaScript **Date objects** represent a single moment in time in a platform-independent format <milliseconds since...>

Zero state: 1.1.1970, 00:00:00

Differences in time are calculated in milliseconds

Date objects translate from milliseconds to understandable date formats

Date Objects with JS - days

```
Date.now() // returns a Number (milliseconds)

const today = new Date() // create a Date object

today.toString() // make it readable

const tomorrow = new Date("2024-11-13") // standardized Date format

tomorrow.toString()

tomorrow.getDay() - today.getDay(); // #dagen
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/Date

<https://stackabuse.com/javascript-get-number-of-days-between-dates/>

Date Objects with JS - time

```
const begin = Date.now()  
  
const end = Date.now()  
  
let elapsed = end - begin // elapsed is a Number (milliseconds)  
  
let secs = // # seconds
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/Date

<https://stackabuse.com/javascript-get-number-of-days-between-dates/>

Date Objects with JS - time

```
const begin = Date.now()

const end = Date.now()

let elapsed = end - begin // elapsed is a Number (milliseconds)

let beginTime = new Date(begin)

let endTime = new Date(end)

let secs = endTime.getSeconds() - beginTime.getSeconds()
```

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date/Date

<https://stackabuse.com/javascript-get-number-of-days-between-dates/>

Date Objects with d3

d3: JS compatible with (far) more options

```
const today = new Date(2023, 11, 21);
```

```
const tomorrow = new Date(2023, 11, 22);
```

```
const days = d3.timeDay.count(today, tomorrow);
```

d3.timeWeek, d3.timeMonth etc.

Time scales!

<https://d3js.org/d3-time>

<https://d3js.org/d3-scale/time>

Pauze

Schedule

1. Date and Time in JS
2. **Spatial Data for the Web**
3. Geospatial Apps



Spatial data

- Spatial data describe anything covering an “area”, on Earth or in space
- a shape or a position
- position on Earth: [latitude, longitude]
- To visualize spatial data, we need: a **map**, the **data** and a **library / Geospatial application**

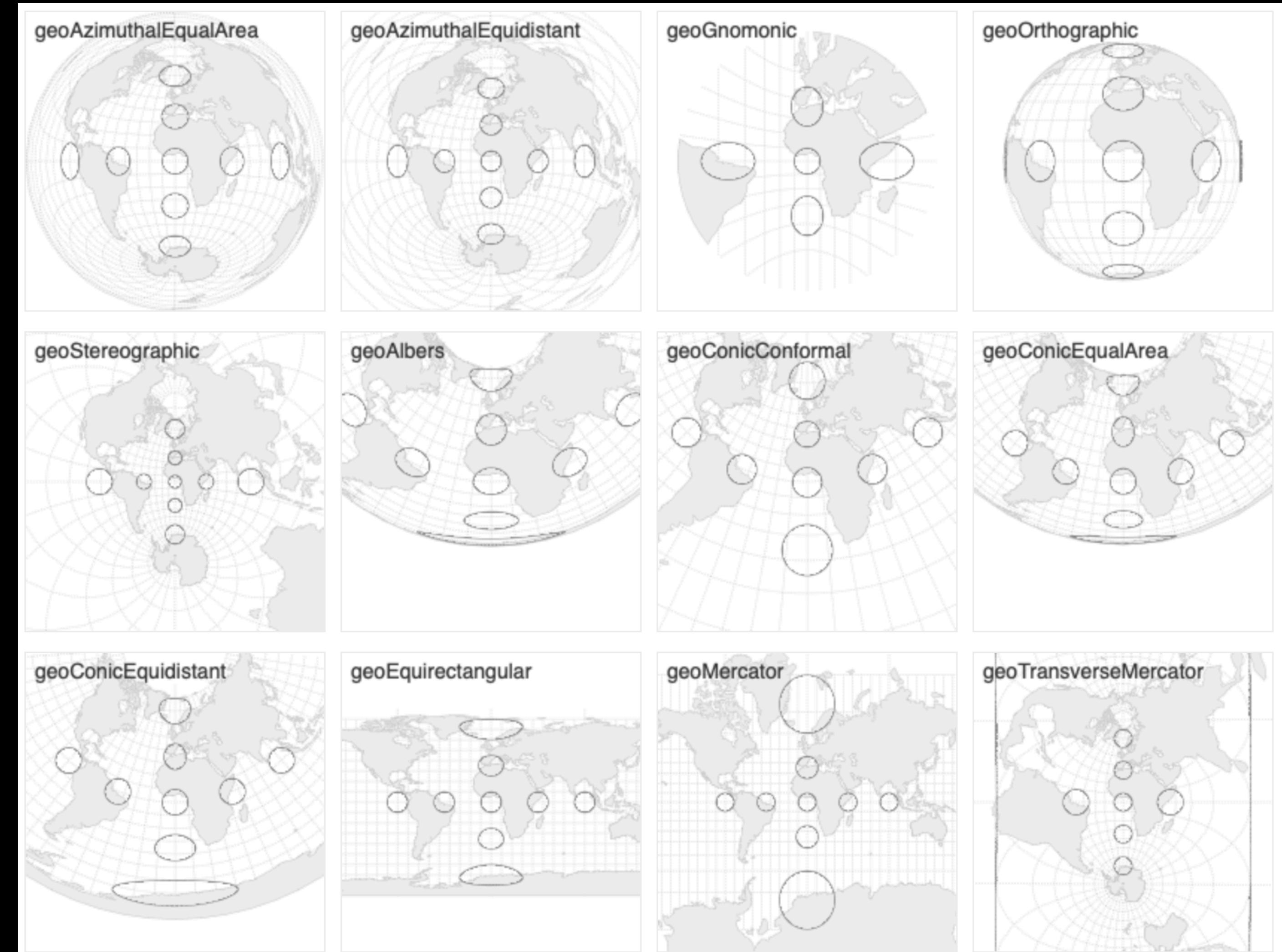
Latitude and longitude finder: <https://www.latlong.net/>

Map projections: a problem

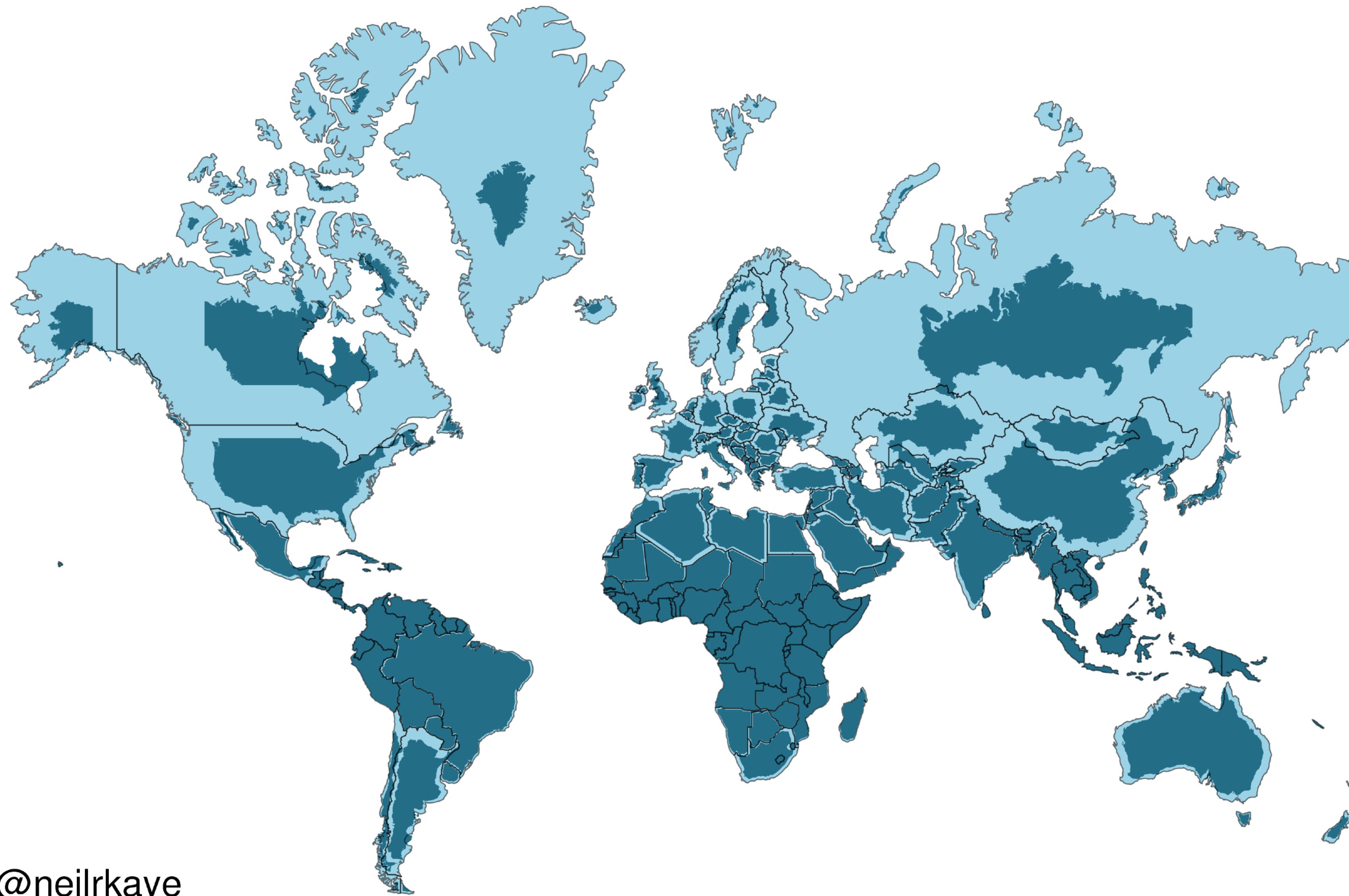
All map projections
are a compromise. Pick
the one that fits with
your dataviz.

See differences here:

[https://
projectionwizard.org/](https://projectionwizard.org/)



World Mercator projection with true country size added



@neilrkaye

Spatial data - formats

Spatial data can be stored as:

- Raster-data (points, lines)
- Vector-data (points, lines, arcs)

Raster data is faster but can have a pixelated look.

Vector data can be rendered more accurately (no grid!) but needs a lot of maintenance to ensure reliability

There are different ways of describing spatial data.

<https://carto.com/blog/raster-vs-vector-whats-the-difference-which-is-best>

W3C on formats for spatial data

	GML	KML	GeoJSON	(HTML)
Based on	XML	XML	JSON	HTML
By		Google		
Use	Spatial Things and 0D-3D geometries	Spatial Things and 0D-3D geometries. Focus on interaction	Spatial Things and 0D-2D geometries	Descriptions of Spatial Things and geometries can be embedded using mechanisms (...) using vocabularies
Tool support	Widely in GIS tools Some Web libraries, 2D usually converted in GeoJSON	Mainly supported by Earth browsers, as Google Earth	Supported in some GIS tools Widely supported non Web libraries and mapping APIs	Optimal for Web publication and discovery
3D-support	Yes	Yes	No	Depends on the vocabulary used (...)

Client-side JS libraries for spatial data

	topoJSON	MapLibreGL	GeoJSON
Based on	Extension of GeoJSON	WebGL, typeScript	JSON
By	Mike Bostok	Open Source, new (> 2022)	
Use	Spatial Things and 0D-2D geometries Polygons, Arcs	Spatial Things and 0D-3D geometries. Interactive maps	Spatial Things and 0D-2D geometries. Polygons
	Vector data (vgl. SVG)	Vector data	Raster data
Used by	D3	Kepler	
3D-support	No	Yes	No

Zie ook: <https://maplibre.org/> , <https://nl.wikipedia.org/wiki/TopoJSON>, <https://nl.wikipedia.org/wiki/GeoJSON>

GeoJSON, TopoJSON

```
{  
  "id": 3,  
  "type": "Feature",  
  "geometry": {  
    "type": "Point",  
    "coordinates": [  
      4.897985,  
      52.3743134  
    ]  
  },  
  "properties": {  
    "Naam": "Oude Kerk 1306",  
    "Functie": "Kerk",  
    "Foto": "Oude kerk.jpg",  
    "Opmerking": "Bron: Cornelis Anthonisz 1544",  
    "Begin": 1306,  
    "Eind": 0  
  }  
}
```

```
{  
  "type": "Topology",  
  "objects": {  
    "example": {  
      "type": "GeometryCollection",  
      "geometries": [  
        {  
          "type": "Point",  
          "properties": {  
            "naam": "Oude Kerk 1306"  
          },  
          "coordinates": [102, 0.5]  
        },  
        {  
          "type": "LineString",  
          "properties": {  
            "naam": "Lijn 53",  
            "omschrijving": "route metrolijn"  
          },  
          "arcs": [0]  
        },  
        {  
          "type": "Polygon",  
          "properties": {  
            "adres": "Wibautstraat 2-4",  
            "naam": {  
              "huidig": "Kohnstammhuis",  
              "voorheen": "Belastingdienst"  
            }  
          },  
          "arcs": [1]  
        }  
      ]  
    }  
  },  
  "arcs": [  
    [[102, 0], [103, 1], [104, 0], [105, 1]],  
    [[100, 0], [101, 0], [101, 1], [100, 1], [100, 0]]  
  ]  
}
```

Spatial data: conclusion

There is no standard format (yet...) for spatial data for the Web

The most important criteria for choosing an ecosystem (= data format, library / geospatial app) for datavisualization are:

- 2D vs 3D
- raster data vs vector data

Most libraries and apps support geoJSON, but geoJSON might be too simple for some geospatial applications. These applications need a more sophisticated ecosystem.

Schedule

1. Date and Time in JS
2. Spatial Data for the Web
3. Geospatial Apps



GeoCoder

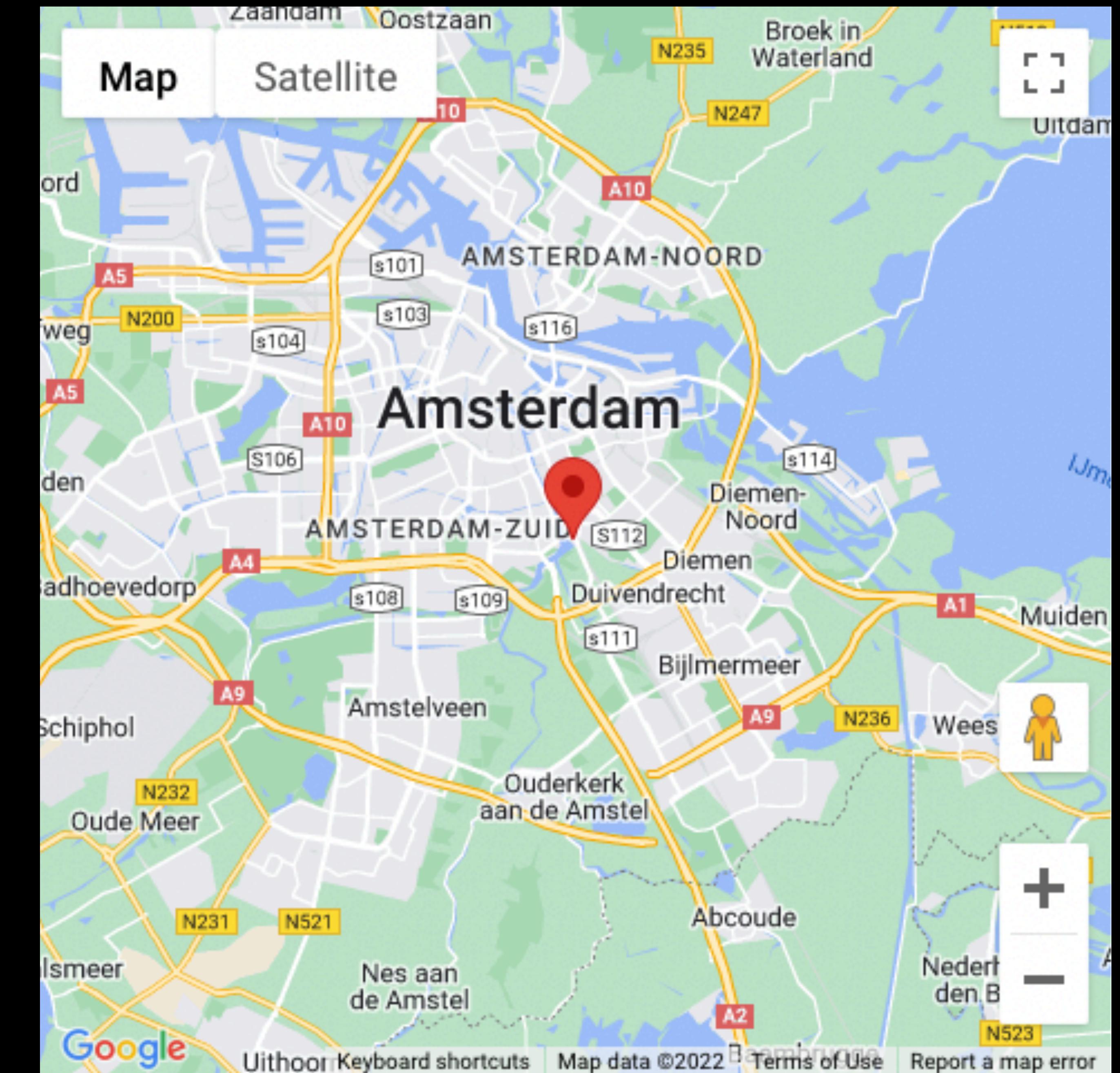
Convert "Wibautstraat 2, Amsterdam"
To 52.35940413649428, 4.908656298091917

Most commercial mapping providers come with their own
geocoder API, but you're only allowed to use them if
you also use their map.

Photon is a free alternative without restrictions: <https://photon.komoot.io/>

<https://www.latlong.net/> , for NL: https://api.pdok.nl/bzk/locatie server/search/v3_1/ui/

Google Maps



Custom Map Styles

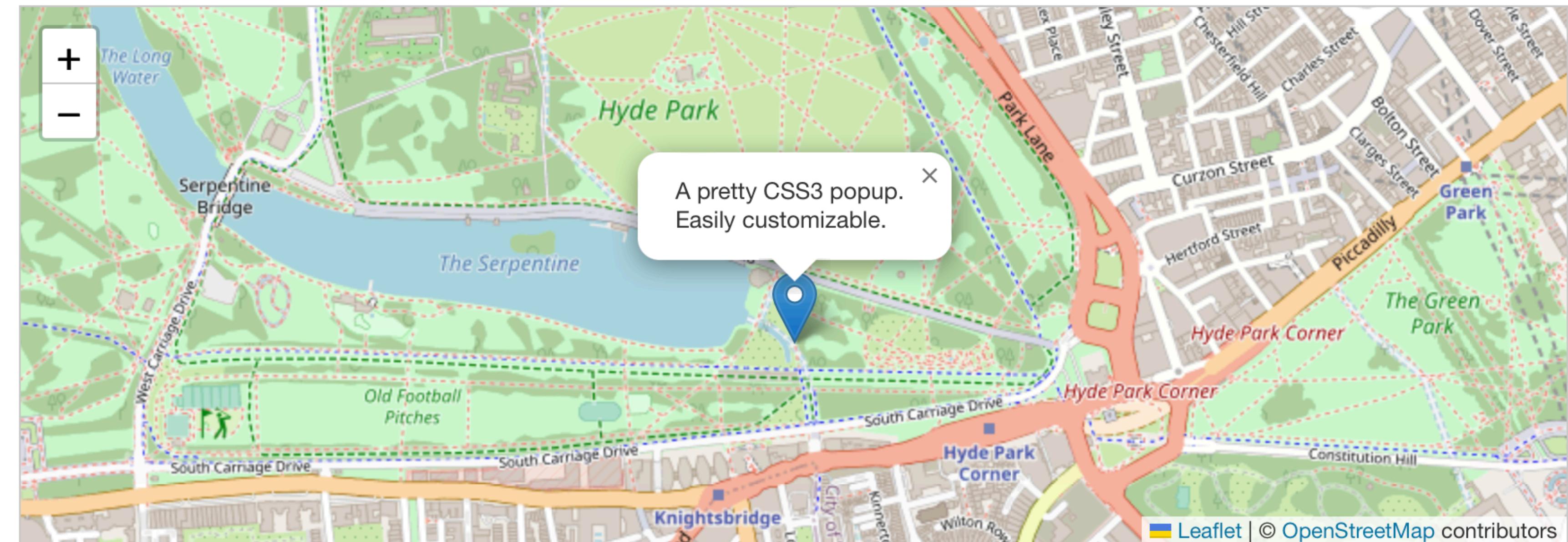


Conditions

- Paid after x number of map requests
- Geocoder only in combination with Google Map
- No caching
- Can't use for internal apps on local network
- Etc.

Leaflet

leafletjs.com



Here we create a map in the 'map' div, add tiles of our choice, and then add a marker with some text in a popup:

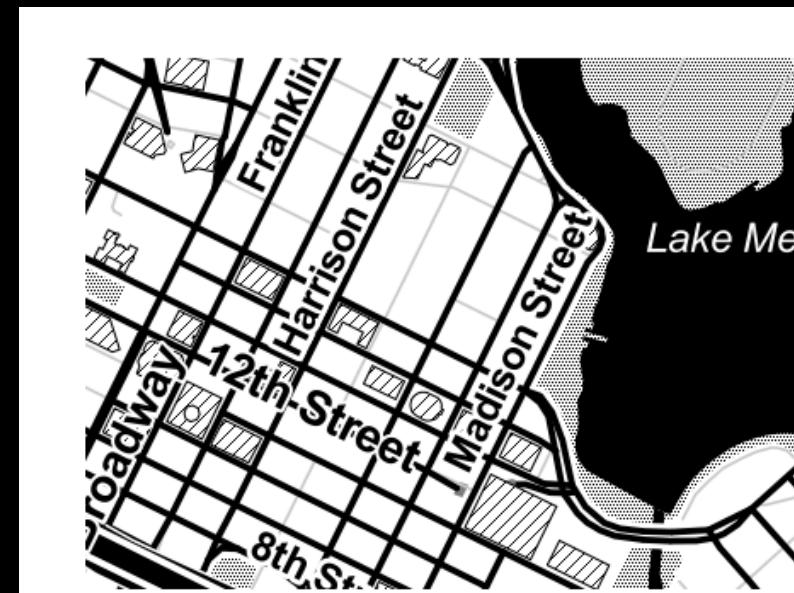
```
var map = L.map('map').setView([51.505, -0.09], 13);

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

L.marker([51.5, -0.09]).addTo(map)
    .bindPopup('A pretty CSS3 popup.<br> Easily customizable.')
    .openPopup();
```

Leaflet: different tile servers (maps)

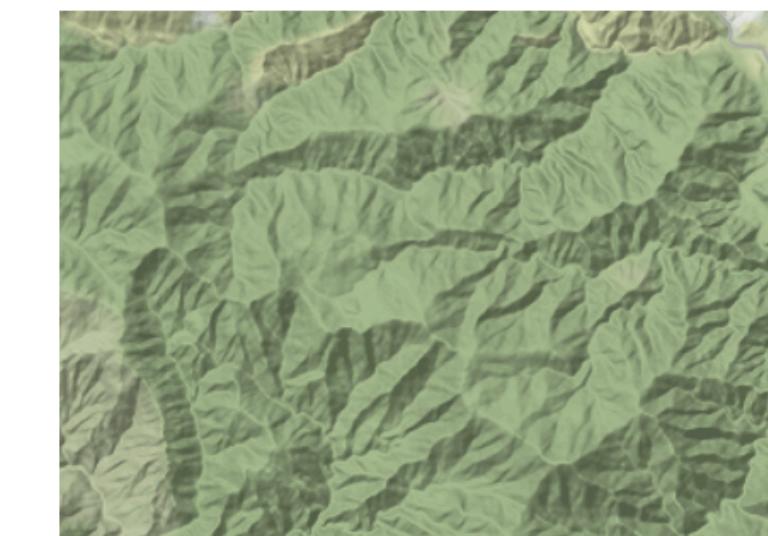
- OpenStreetMap
- MapBox
- HERE
- Stamen ->
- Your own?
- NLMaps



Toner

These high-contrast B+W (black and white) maps are featured in our Dotspotting project. They are perfect for data mashups and exploring river meanders and coastal zones. Available in six flavors: [standard toner](#), [hybrid](#), [labels](#), [lines](#), [background](#), and [lite](#).

Available worldwide.



Terrain

Orient yourself with our terrain maps, featuring hill shading and natural vegetation colors. These maps showcase advanced labeling and linework generalization of dual-carriageway roads. Terrain was developed in collaboration with Gem Spear and Nelson Minar.

Available in four flavors: [standard terrain](#), [labels](#), [lines](#), and [background](#).

Available worldwide.



Watercolor

Reminiscent of hand drawn maps, our watercolor maps apply raster effect area washes and organic edges over a paper texture to add warm pop to any map. Watercolor was inspired by the [Bicycle Portraits project](#). Thanks to [Cassidy Curtis](#) for his early advice.

Available worldwide.

NL Maps



The background of the page features a faint, semi-transparent map of the Netherlands. Overlaid on the top left is the NL Maps logo, which consists of a white grid icon followed by the text "NL Maps". The main title "Dé officiële kaart van Nederland" is centered in large, white, sans-serif font. Below it, a subtitle "De meest actuele kaart nu beschikbaar" is displayed in a smaller, white, sans-serif font. At the bottom center are two purple rounded rectangular buttons. The left button contains the text "Gebruik NL Maps nu" next to a white map pin icon. The right button contains the text "Bekijk op GitHub" next to a white GitHub logo icon.

Dé officiële kaart van Nederland

De meest actuele kaart nu beschikbaar

Gebruik NL Maps nu

Bekijk op GitHub

<https://nlmaps.nl/>

D3



<https://www.d3indepth.com/geographic/>

Summary

	Google Maps	Leaflet / OSM	D3
Price	Paid, \$ x free credits/month	Mostly free (depends on your map provider)	Completely free
Restrictions	Short selection from their ToS: no caching/downloading data, no applications targeting kids, no in-car systems, can't use APIs separately, no text-to-speech	Depends on your map provider, but plenty of options with few restrictions + you can host your own	None
Flexibility	Can be styled with your own colours + choose how much info is shown. Can't pick projections or focus on one part of map	Very flexible options for styling. Can't pick your own projection or focus on one part of the map.	Extremely flexible. You can pick your own projection + bring your own map files and show only one province
Use when	When street-level details are important or when you need access to their POI data	When street-level details are important but you don't need Google's POI data	When you want more flexible ways to visualise your data and don't need street-level detail

**Uncaught SyntaxError
Unexpected end of input**