DATA VISUALIZATION AND VISUAL ANALYTICS

S. Rinzivillo – rinzivillo@isti.cnr.it

CARTOGRAPHY IN D3

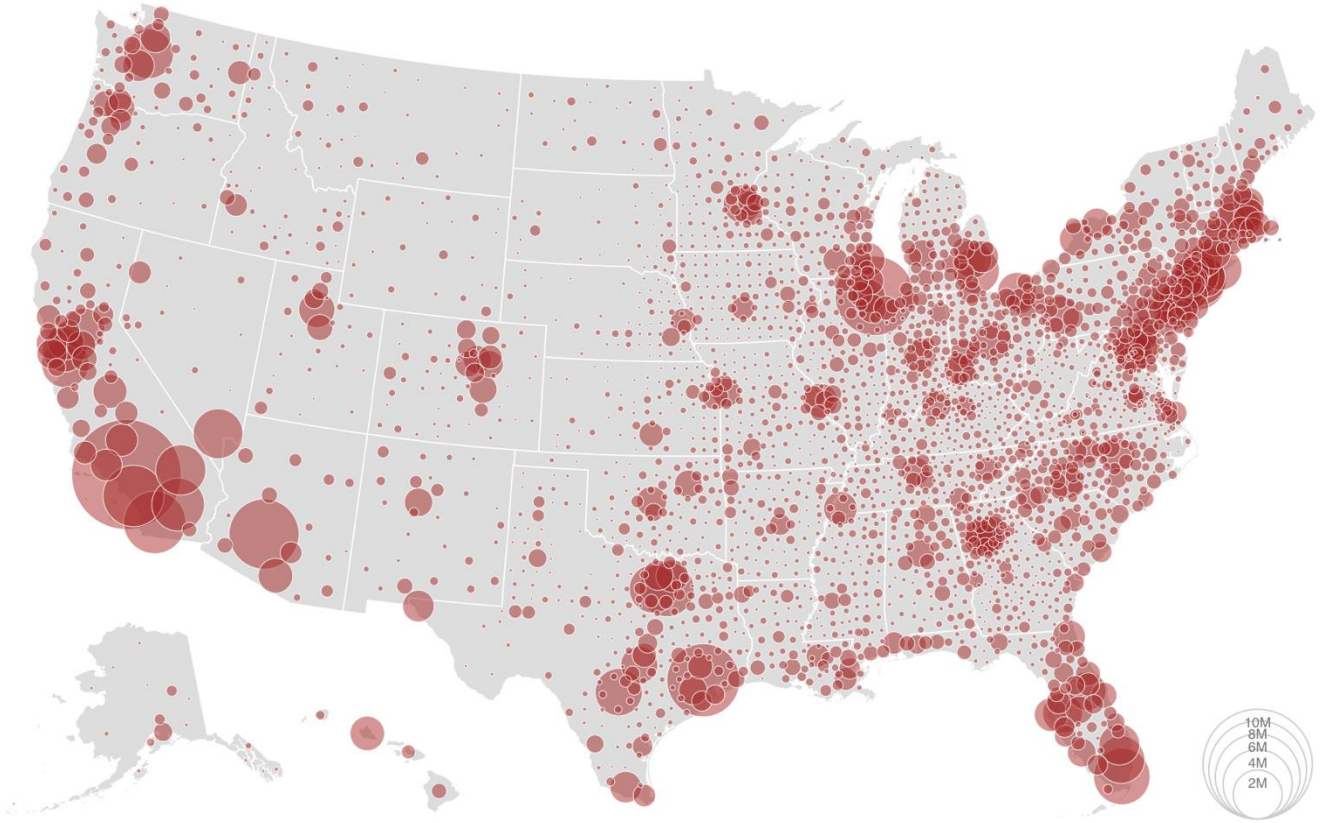
GEOGRAPHICAL DATA- GEOJSON

- Javascript based description of geographical objects
- Three main properties:
  - type (e.g. “FeatureCollection”)
  - crs (Coordinate Reference System)
  - features (list of objects located in the space)
- Each feature consists of:
  - type (e.g.“Feature”)
  - properties (key-value dictionary of attributes)
  - geometry (geometry that represents the object)

FROM GEO-DATA TO VISUALIZATION

- Mapping of geographical coordinates (longitude and latitude) to position on the screen
  - Definition of projection function based on scale, translation and rotation
  - module: d3.geo.projection
- Creation of mapping for each element (point) of the geometry
  - module: d3.geo.path

EXAMPLE: MAP OF PAINTINGS



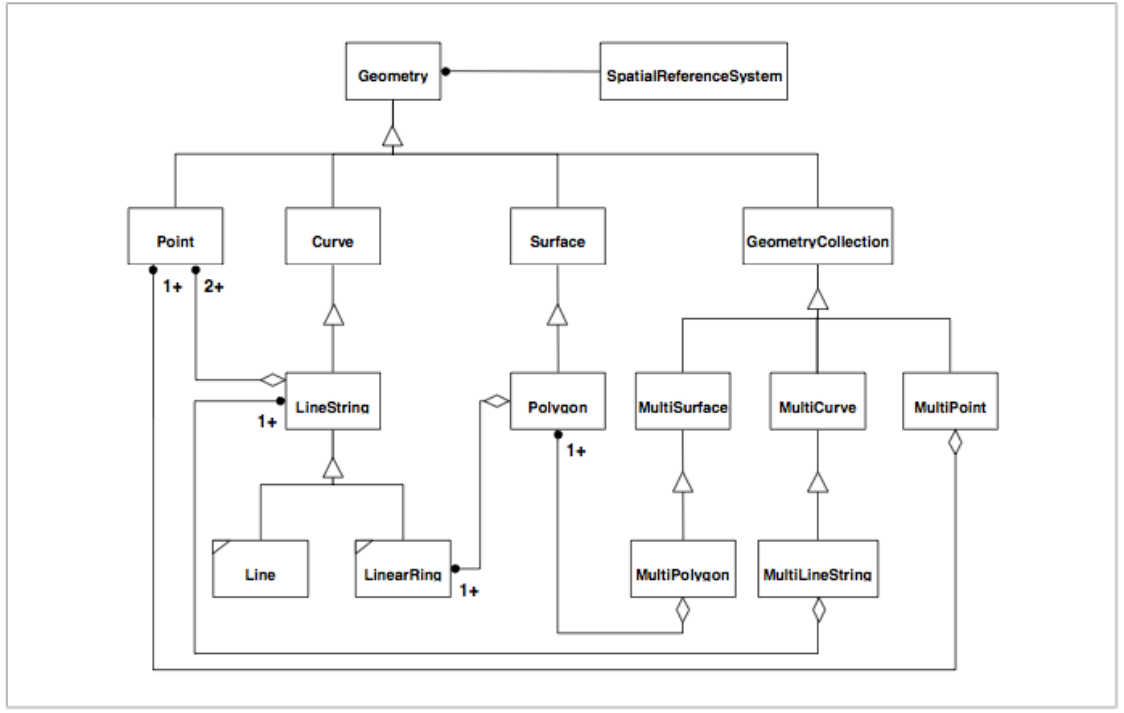
GEOMETRIES AND STANDARDS

OPEN GIS CONSORTIUM – OGC

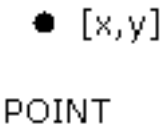
- Consortium to define protocols to transmitt territorial and geographic information

SFS – SIMPLE FEATURE SPECIFICATION

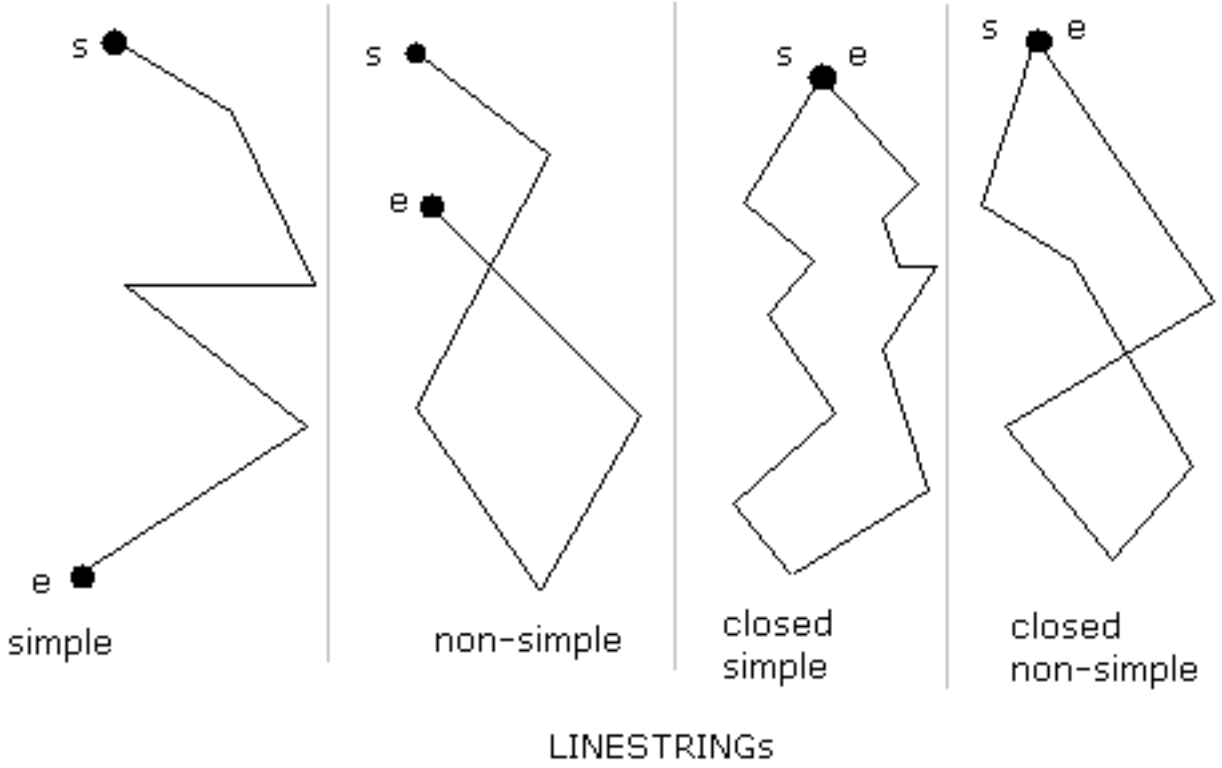
- Definition of an abstract data type:
  - Geometry



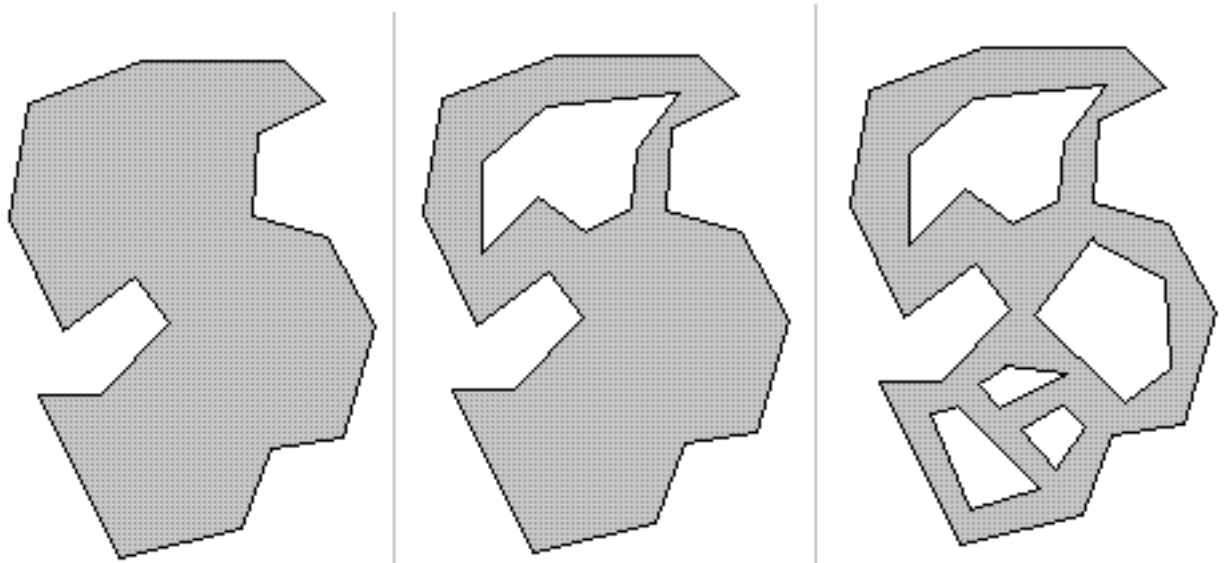
POINT



LINESTRING

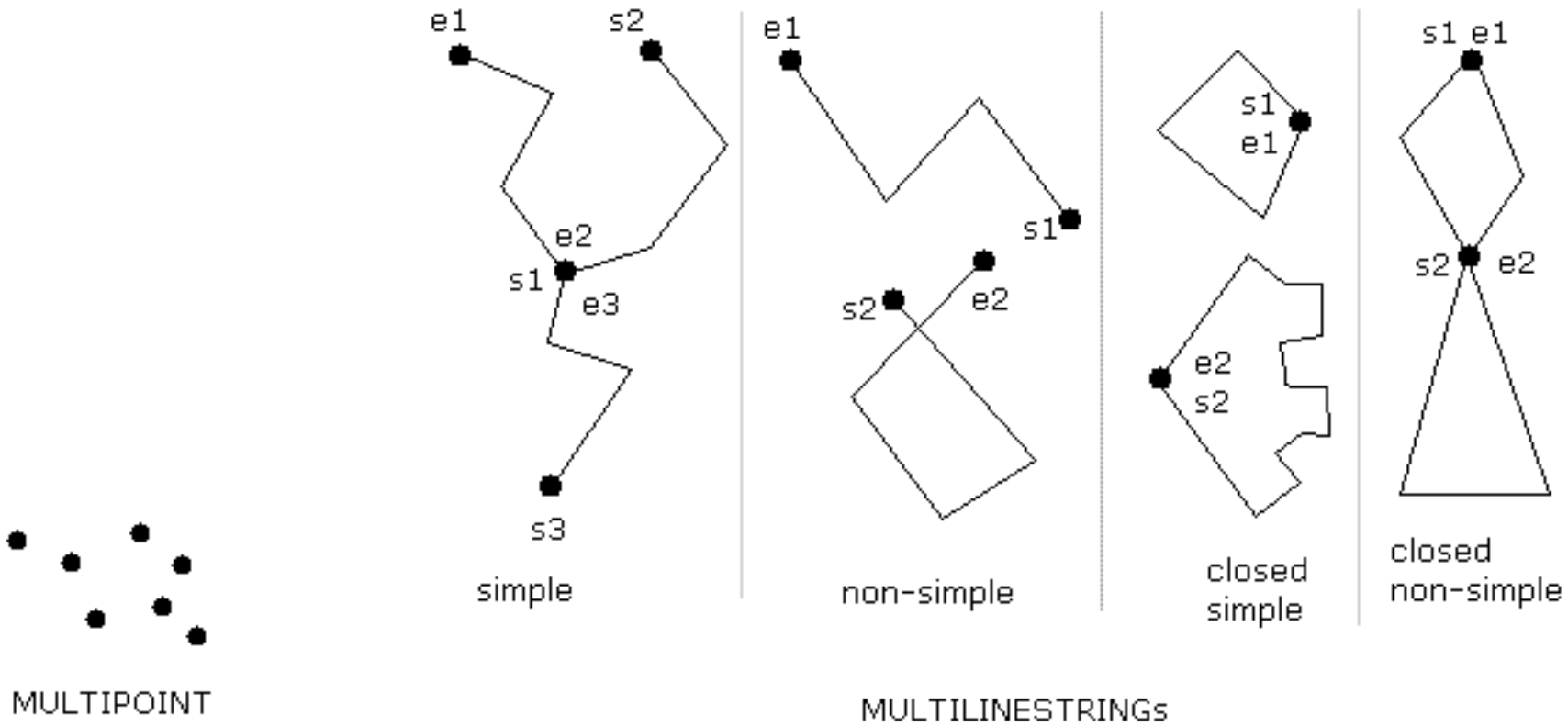

s
e
simple
s
e
non-simple
s
e
closed
simple
s
e
closed
non-simple
LINESTRINGs

POLYGON

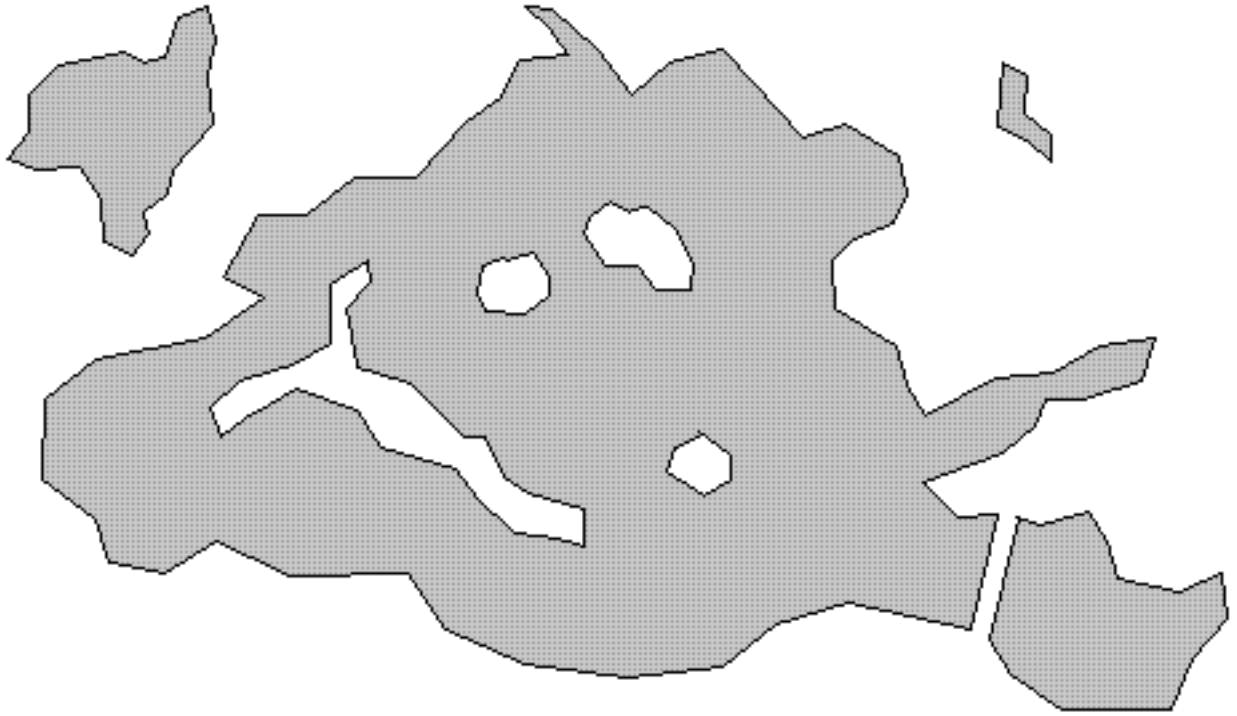


POLYGONs

MULTIPOINT, MULTILINESTRING



MULTIPOLYGON



MULTIPOLYGON

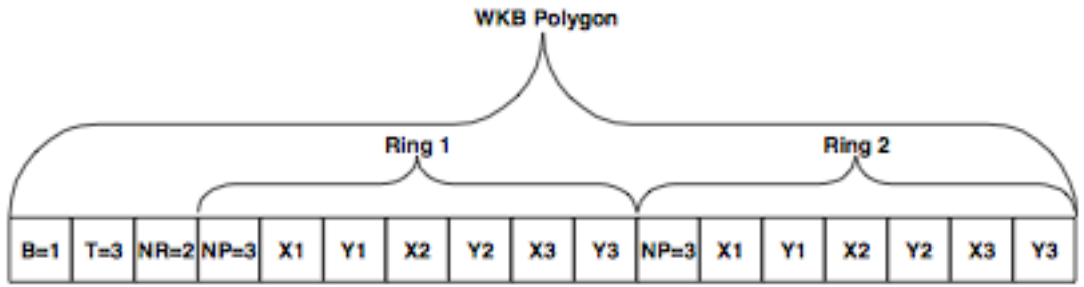
WKT – WELL KNOWN TEXT FORMAT

- POINT(123.45 543.21)
- LINESTRING(100.0 200.0, 201.5 102.5, 1234.56 123.89)
- POLYGON((101.23 171.82, 201.32 101.5, 215.7 201.953, 101.23 171.82))
- POLYGON((10 10, 20 10, 20 20, 10 20, 10 10),(13 13, 17 13, 17 17, 13 17, 13 13))
- MULTILINESTRING((1 2, 3 4), (5 6, 7 8, 9 10), (11 12, 13 14))
- MULTIPOLYGON(((0 0,10 20,30 40,0 0),(1 1,2 2,3 3,1 1)),((100 100,110 110,120 120,100 100)))
- GEOMETRYCOLLECTION(POINT(1 1), LINESTRING(4 5, 6 7, 8 9), POINT(30 30))

WKB – WELL KNOWN BINARY FORMAT

- Compact representation
- Useful to store geometries on DBMS

```
WKBPolygon {
        byte        byteOrder;
        uint32      wkbType;                    // 3
        uint32      numRings;
        LinearRing  rings[numRings];
}
```



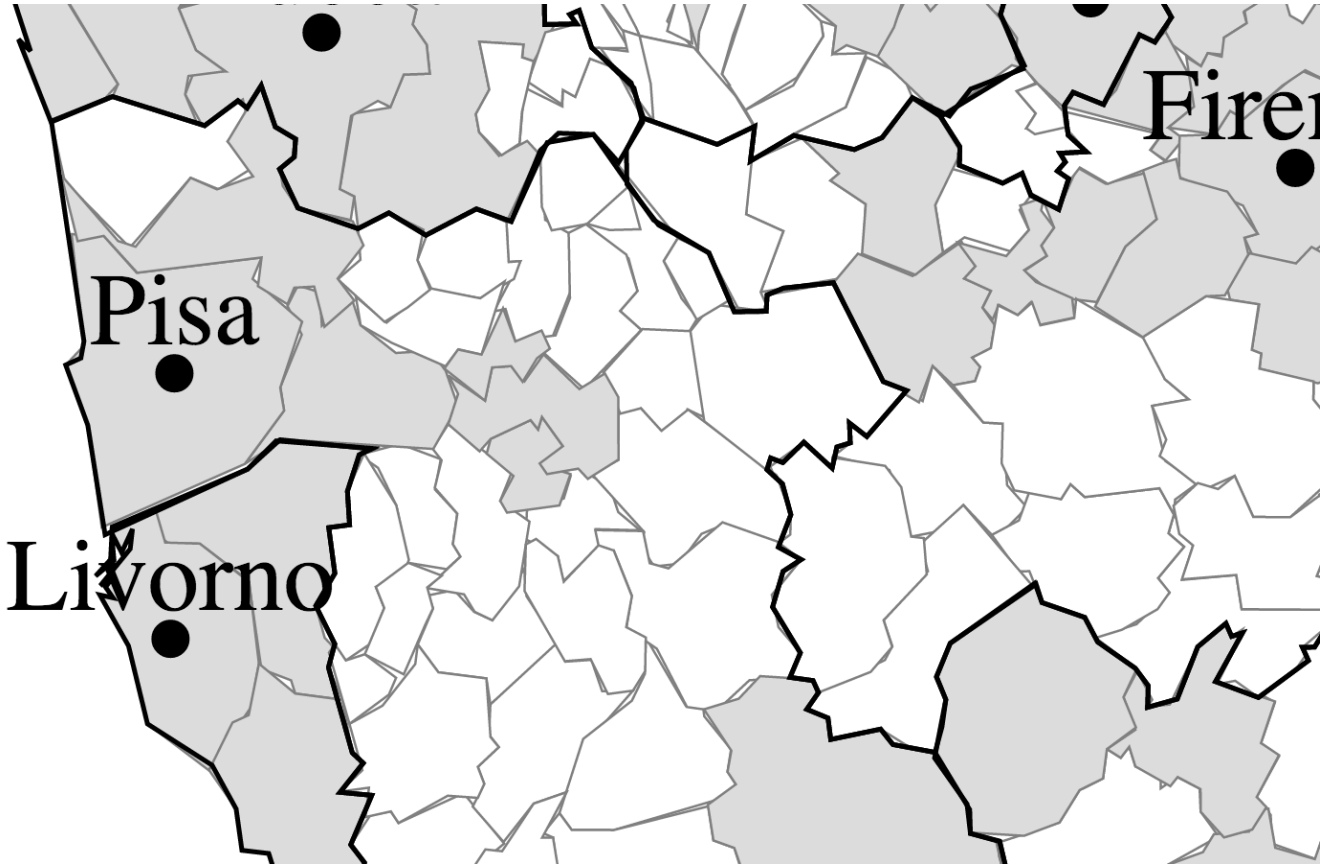
GEOJSON FORMAT

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {"type": "Point", "coordinates": [102.0, 0.5]},
      "properties": {"prop0": "value0"}
      },
    { "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [102.0, 0.0], [103.0, 1.0], [104.0, 0.0], [105.0, 1.0]
          ]
        },
      "properties": {
        "prop0": "value0",
        "prop1": 0.0
        }
      },
    { "type": "Feature",
      "geometry": {
        "type": "Polygon",
        "coordinates": [
          [ [100.0, 0.0], [101.0, 0.0], [101.0, 1.0],
            [100.0, 1.0], [100.0, 0.0] ]
          ]
      },
      "properties": {
        "prop0": "value0",
        "prop1": {"this": "that"}
        }
      }
    ]
  }
```

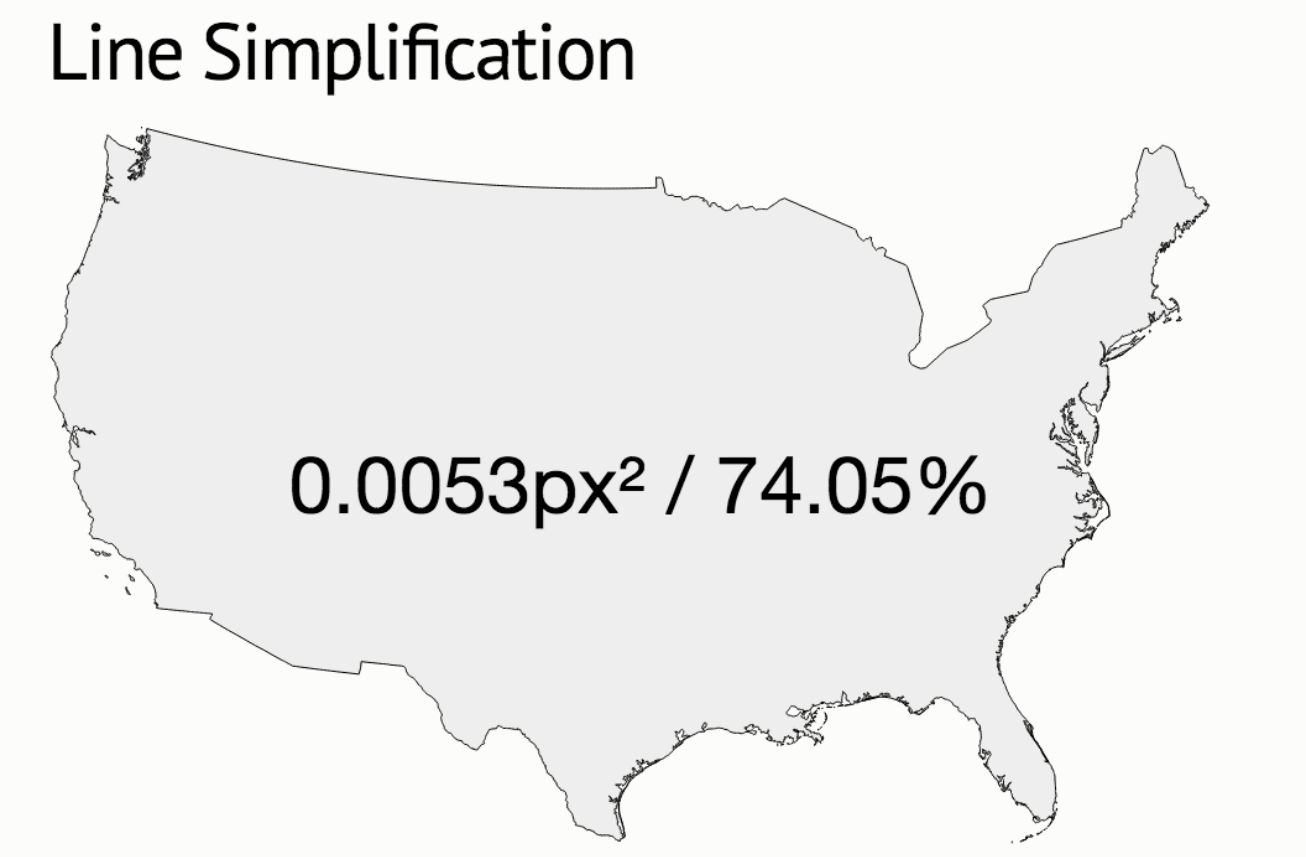
TOPOJSON

- Extends GeoJSON and encodes topology
- Shared lines are represented as arcs
- Reduce redundancy and decrease file size
- Topology can be exploited in specific applications

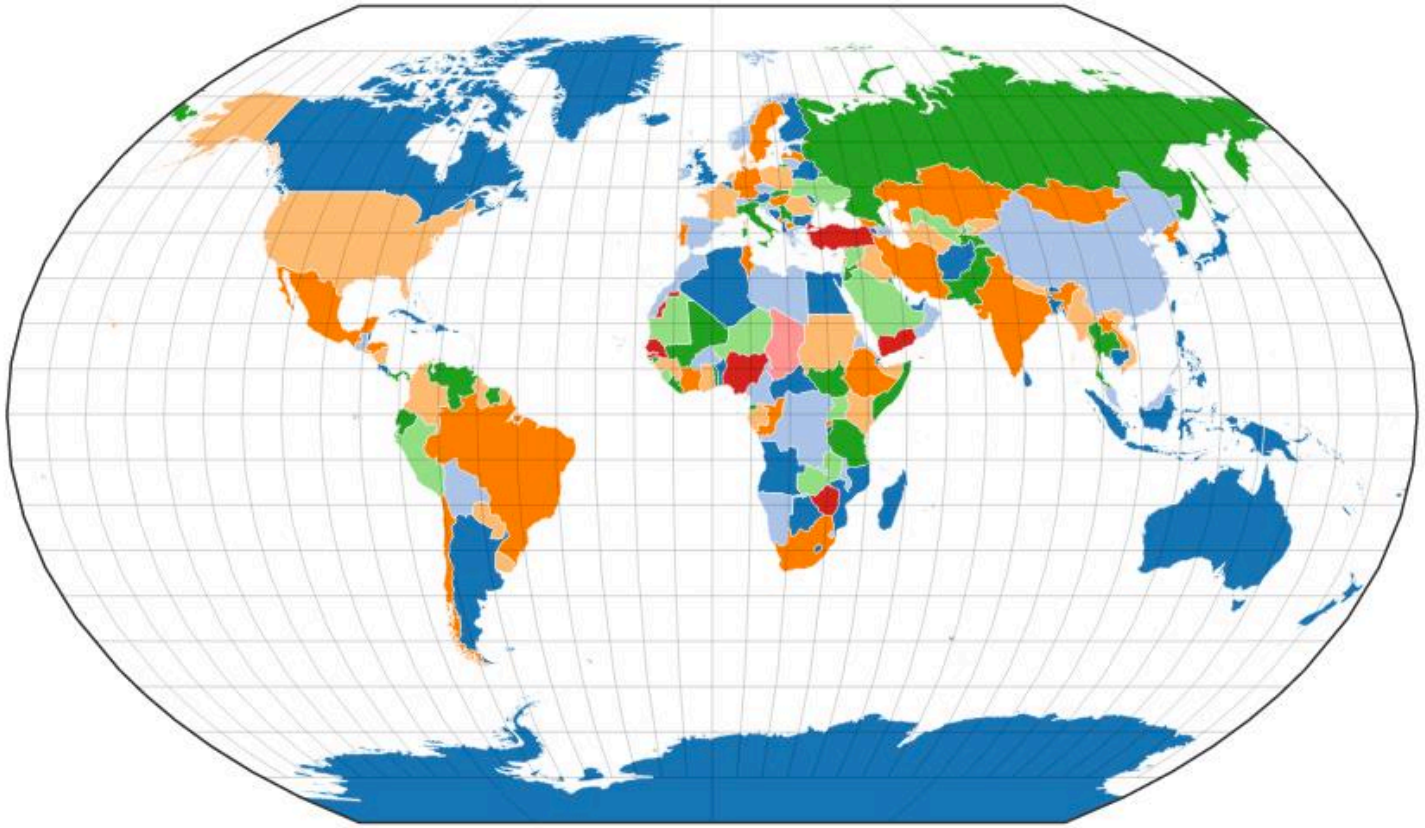
EXAMPLE, WITHOUT TOPOLOGY



LINE SIMPLIFICATION

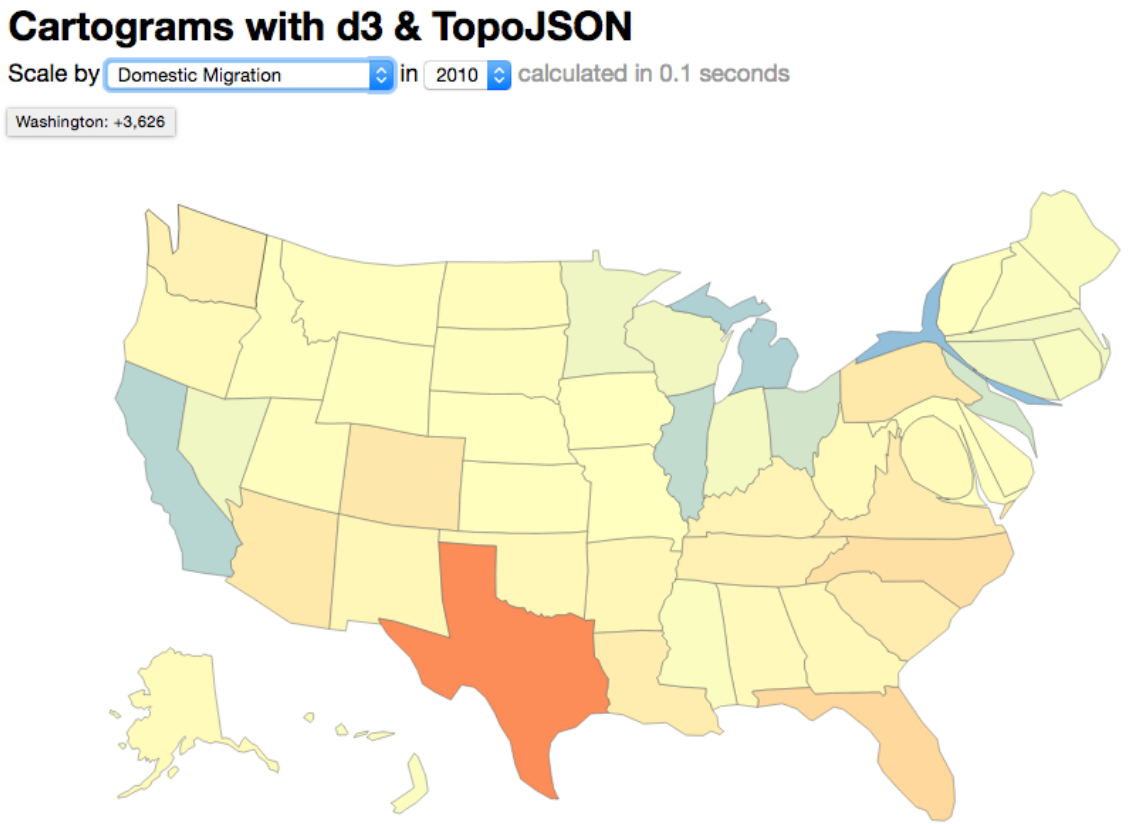


COLOR MAPPING

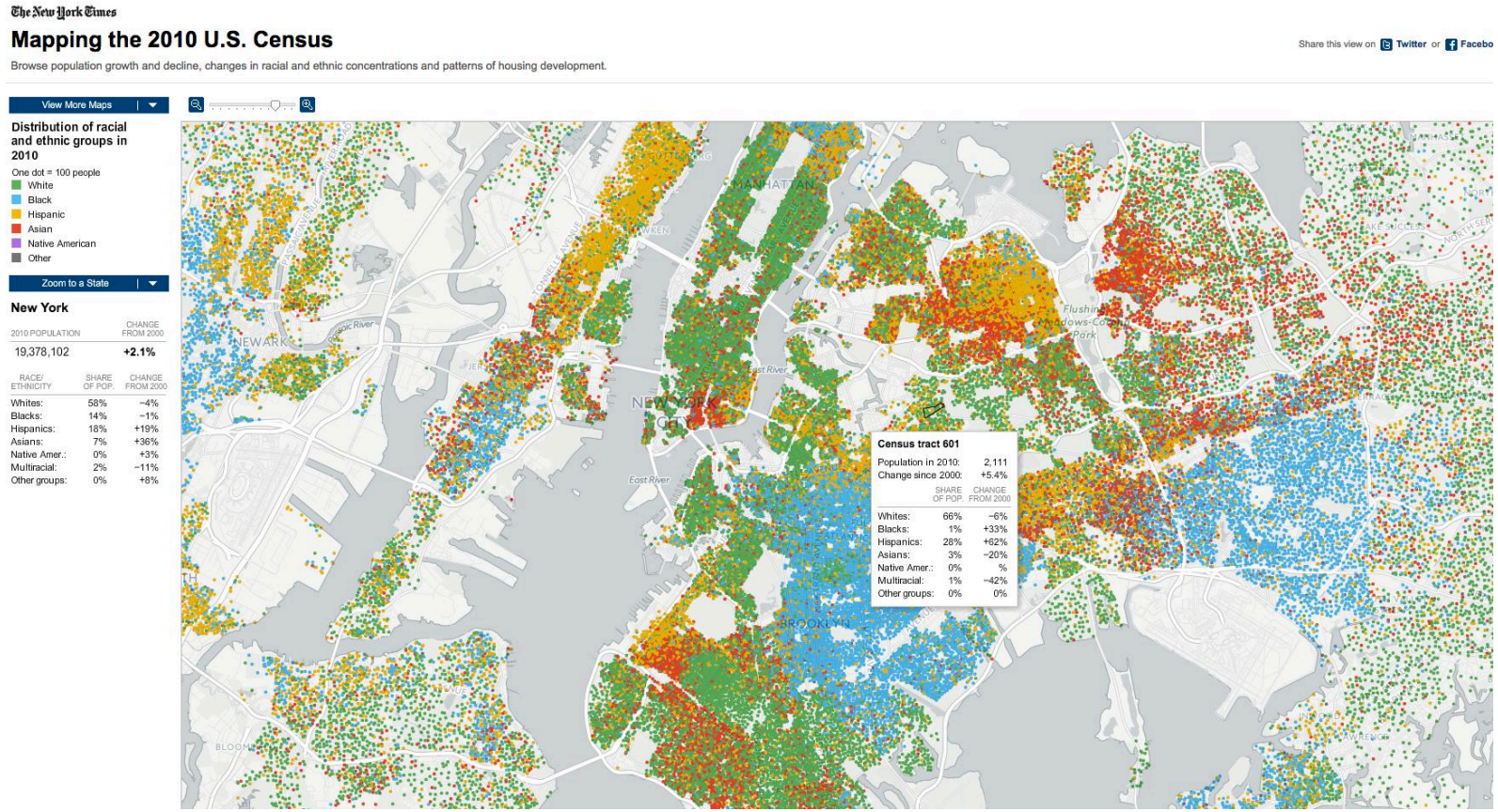


http://bl.ocks.org/jasondavies/4188334

CARTOGRAMS

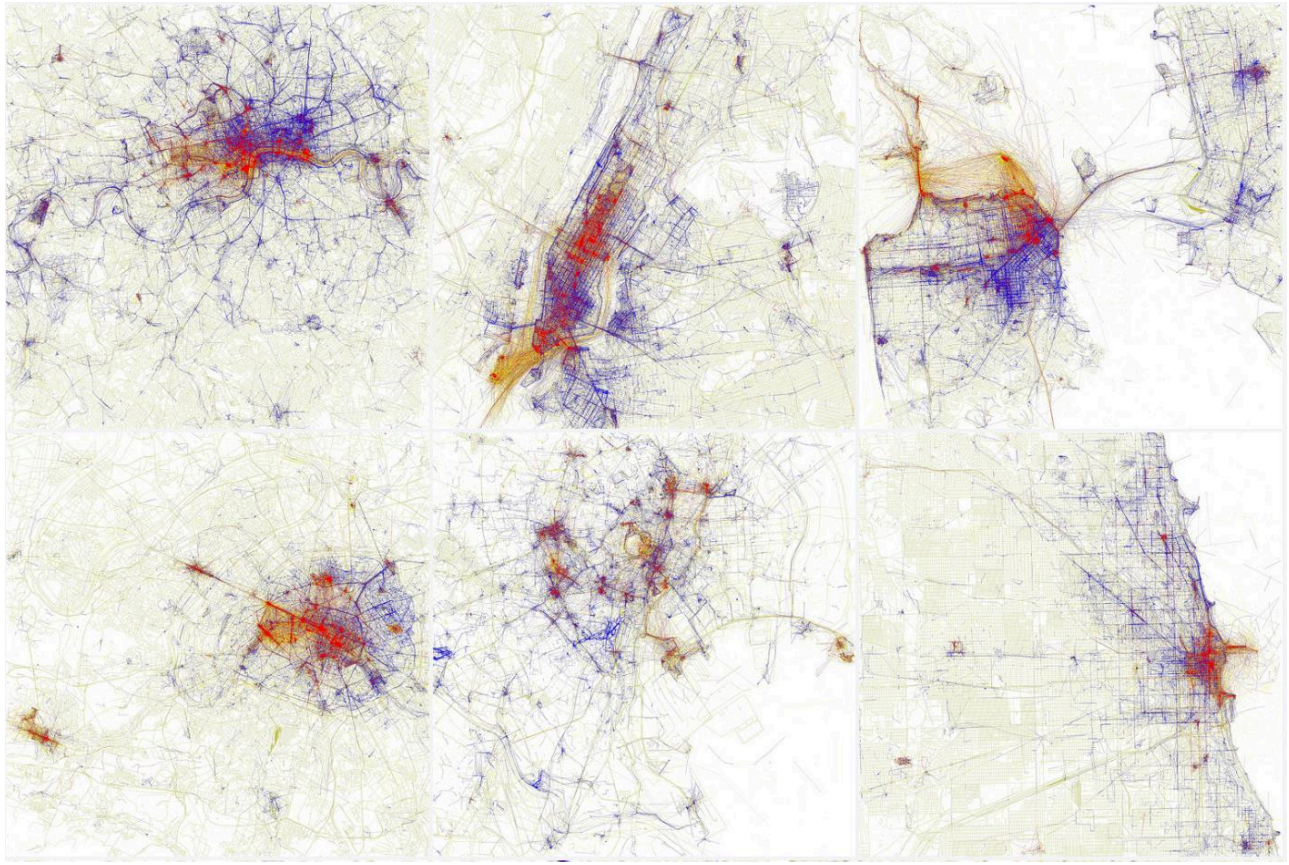

Cartograms with d3 & TopoJSON
Scale by Domestic Migration in 2010 calculated in 0.1 seconds
Washington: +3,626

DOT DISTRIBUTION



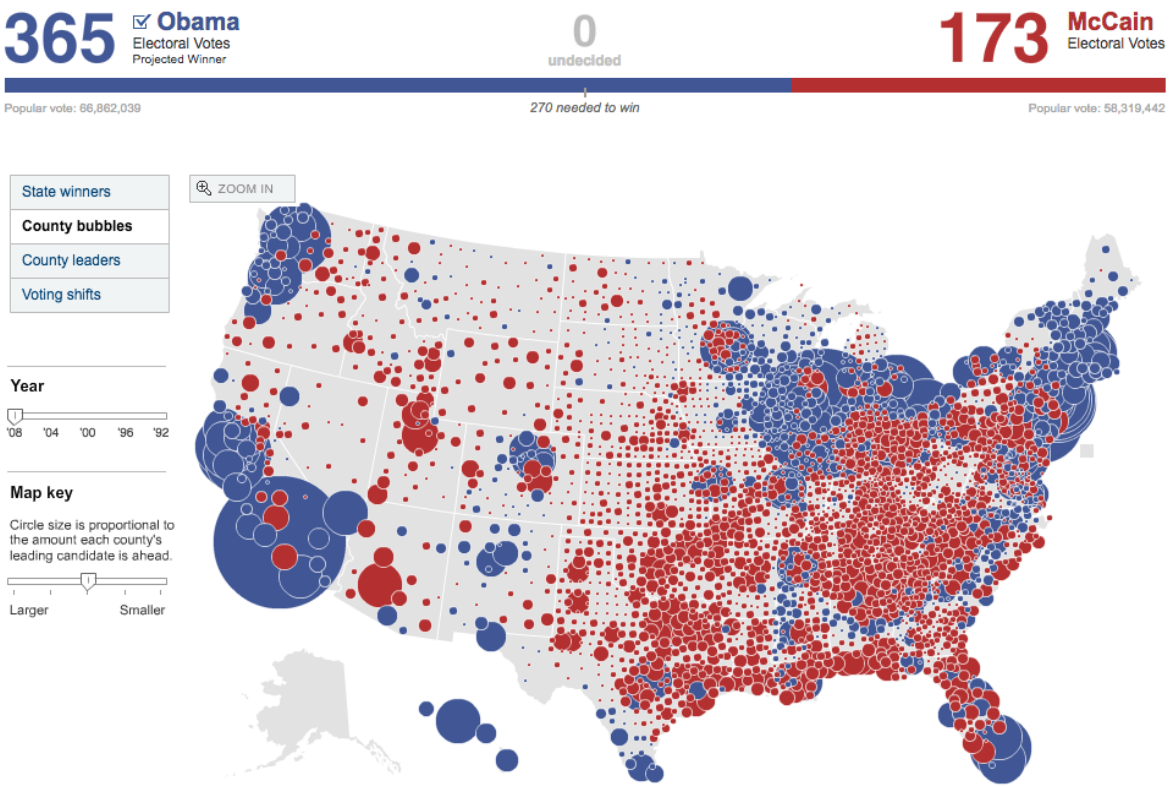
http://projects.nytimes.com/census/2010/map

LINES DISTRIBUTION



https://www.flickr.com/photos/walkingsf/albums/72157624209158632

GRADUATED SYMBOL MAP



TILE MAP SERVER

TILE MAP SERVER

- An efficient solution to publish maps on the web
  - Complexity in space (rather than in time)
  - Used by many providers:
    - Google Maps, Bing, Yahoo Maps, OpenStreetMaps
- Maps is generated once for all level of zoom and then sliced into tiles
- A map for a finite set of zoom levels

TILE MAP SERVER (2)

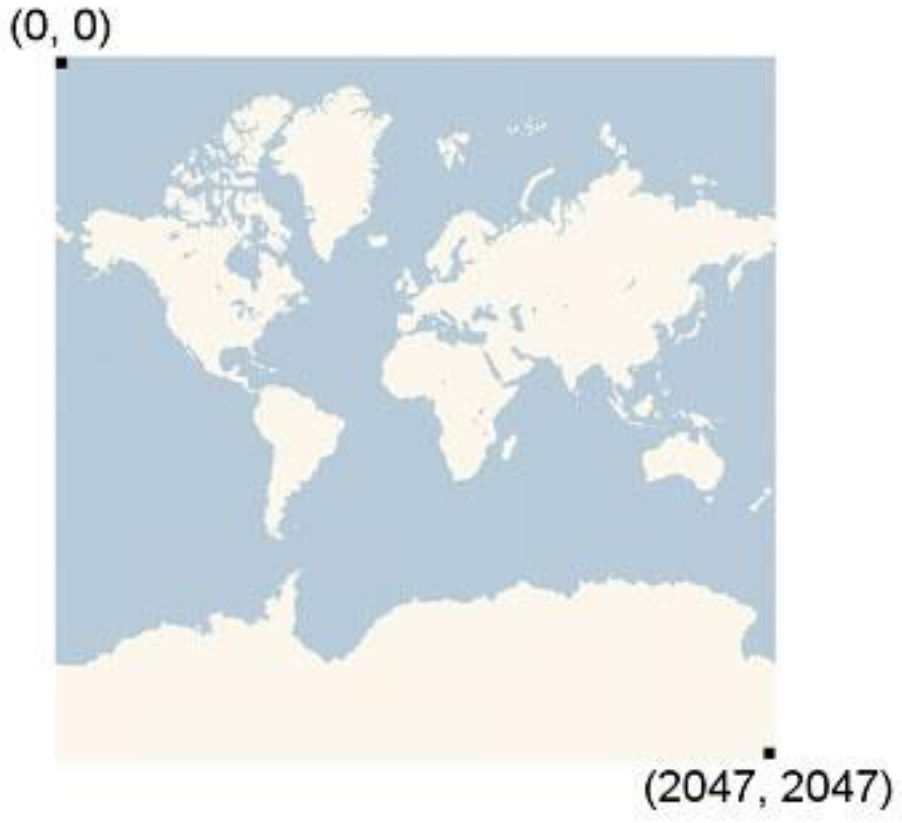
- To simplify coordinate mapping: cylindrical projection
- Two main reference systems:
  - Sphere Mercator (53004)
  - World Mercator (54004)
- Mercator Cyndric projection
  - Meridians are parallels
  - Conformal (preserves shapes)
  - Preserves directions

TILE MAP SERVER (3): SCALE

- Hierarchy division of plane
- Every tile (any zoom) has a fixed dimension: 256x256
- Each zoom level increases (doubles) the number of tiles
- At level 1: only 4 tiles

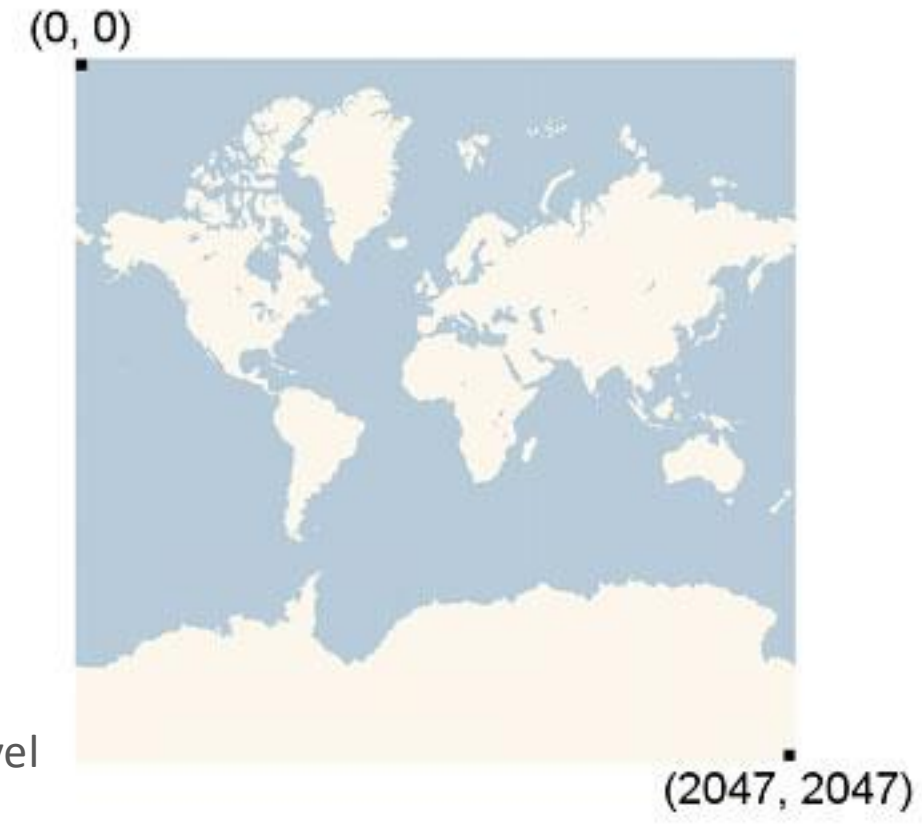
TILE MAP SERVER (4): SCALE

- At each zoom level, the number of tiles increases
  - Every tile at level n generates 4 tiles at level n+1
- At level n tiles cover 256 * 2^n pixel
- For example, at level 3 map has a side 256 * 2^3 = 2048 pixel



TILE MAP SERVER (5): COORDINATES

- Given coordinate (lat,lon) and zoom level n, how to determine position on the image?
- Which tile correspond to coordinate?



pixelX = ((longitude + 180) / 360) * 256 * 2^level

pixelY = (1 – log(tan(Latitude)+ sinh(Latitude)) / pi))/2 * 256 * 2 ^ level

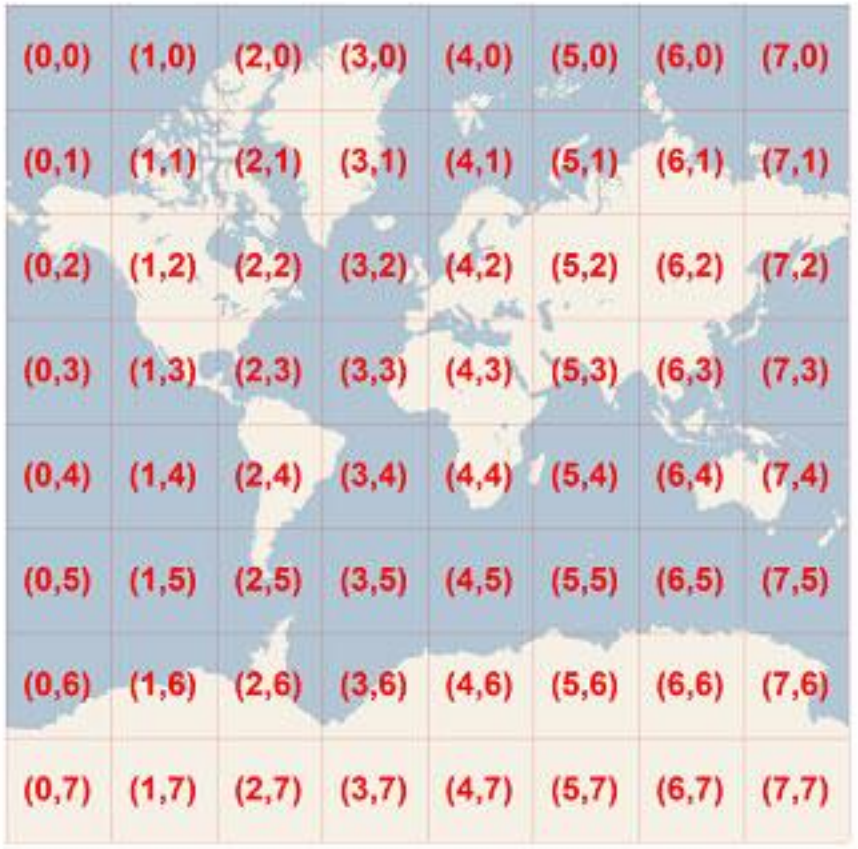
TILE MAP SERVER (6): TILE NUMBER

- Given pixelX e pixelY
- Which tile contains that pixel?

tileX = floor(pixelX / 256)

tileY = floor(pixelY / 256)

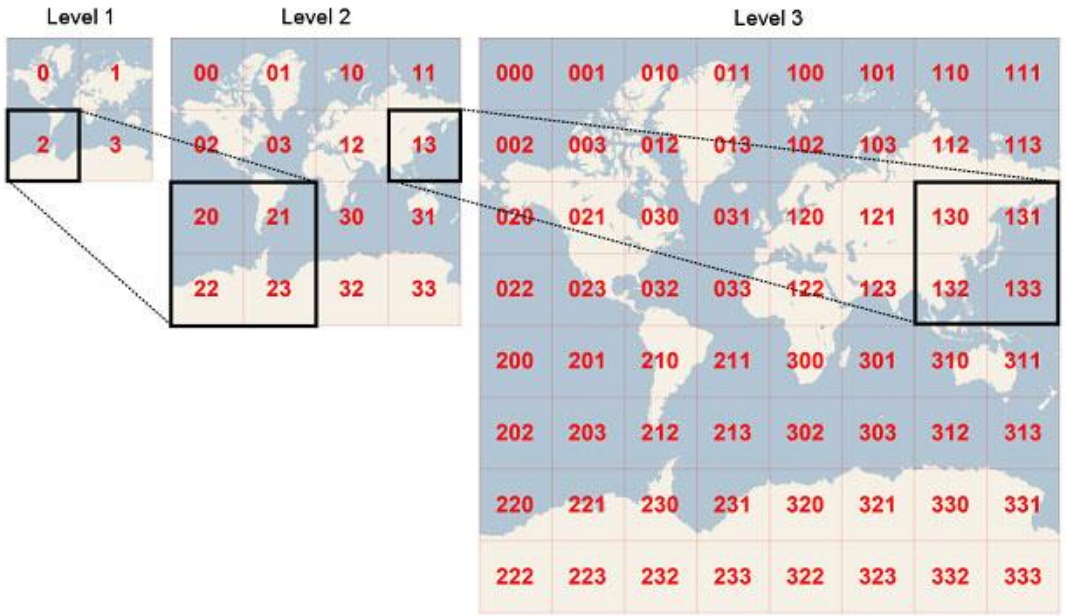
- URL to tile:
  - /zoom/tx/ty
  - quadkey



http://otile2.mqcdn.com/tiles/1.0.0/osm/1/0/0.png

TILE MAP SERVER (7): QUADKEY

- Used by Bing
- Length of the key corresponds to the zoom level



tileX = 3 = 011
tileY = 5 = 101
quadkey = 100111 = 2134 = “213”

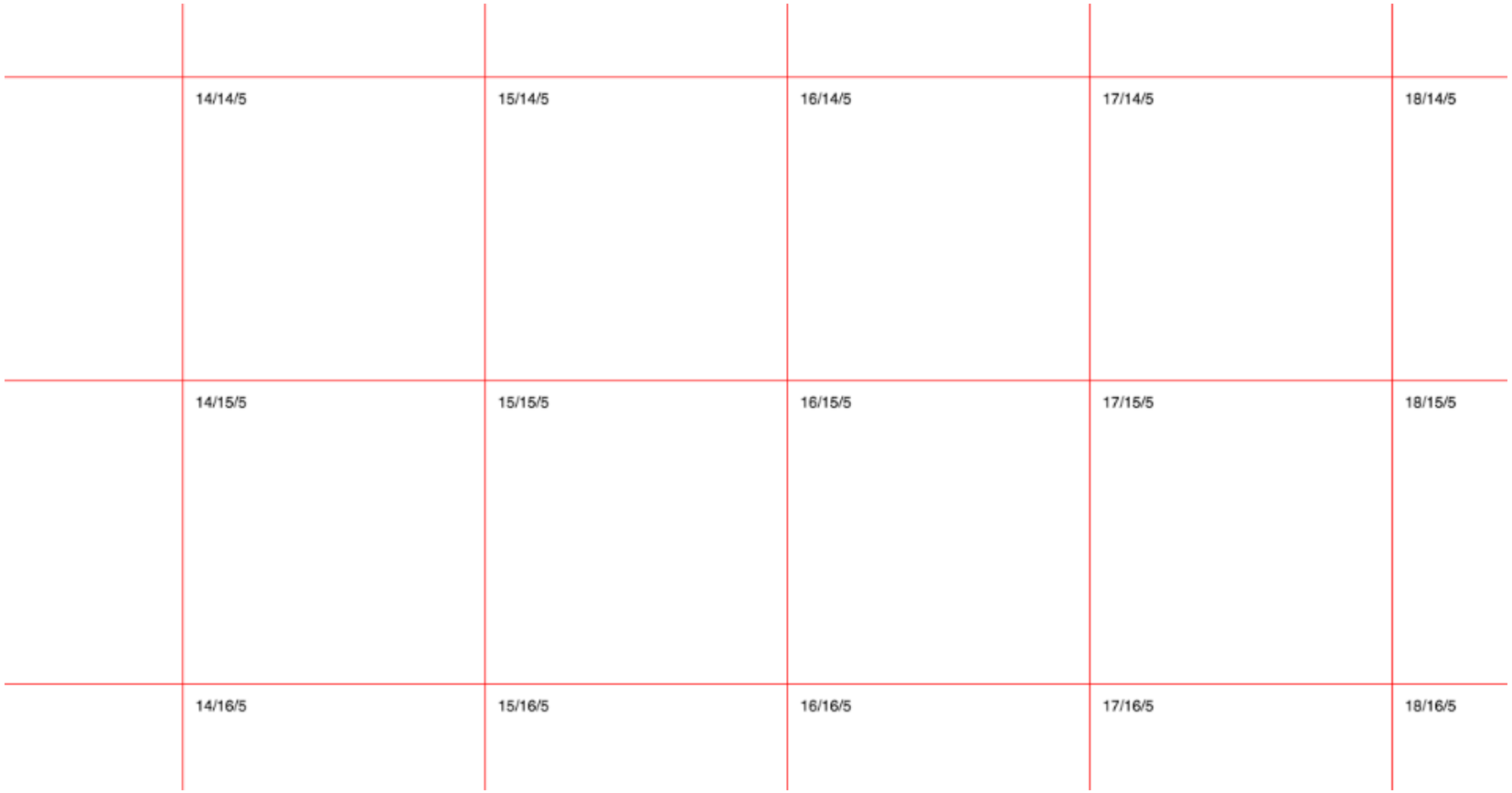
TILE MAP SERVER (8): ZOOM IN

- Given a tile a zoom level n
- Successive tile at level n+1 are:
  - 2x,2y
  - 2x+1,2y
  - 2x,2y+1
  - 2x+1,2y+1

/3/4/2.png


Norway
Sweden
Finland
Bergen
Oslo
Stockholm
Kristiansand
Helsinki
St. Petersburg
Petrozavodsk
Archangel
Vologda
Yaroslavl
Tver
Moscow
Nizhny Nov
Estonia
Latvia
Lithuania
North Sea
Denmark
Malmo
Baltic Sea
Hamburg
Berlin
Gdansk
Netherlands
Poland
Germany
Belgium
Luxembourg
Czech Rep.
Kraków
Munich
Slovakia
Austria
Hungary
France
Switzerland
Slovenia
Croatia
Serbia
Kosovo
Albania
Italy
Adriatic Sea
Marseille
Barcelona
Naples
Belarus
Smolensk
Bryansk
Brest
Voronezh
Kiev
Lvov
Kharkiv
Ukraine
Volgog
Rostov
Moldova
Romania
Krasnodar
Sevastapol
Constanta
Naltchik
Black Sea
Bulgaria
Samsun
Georgia

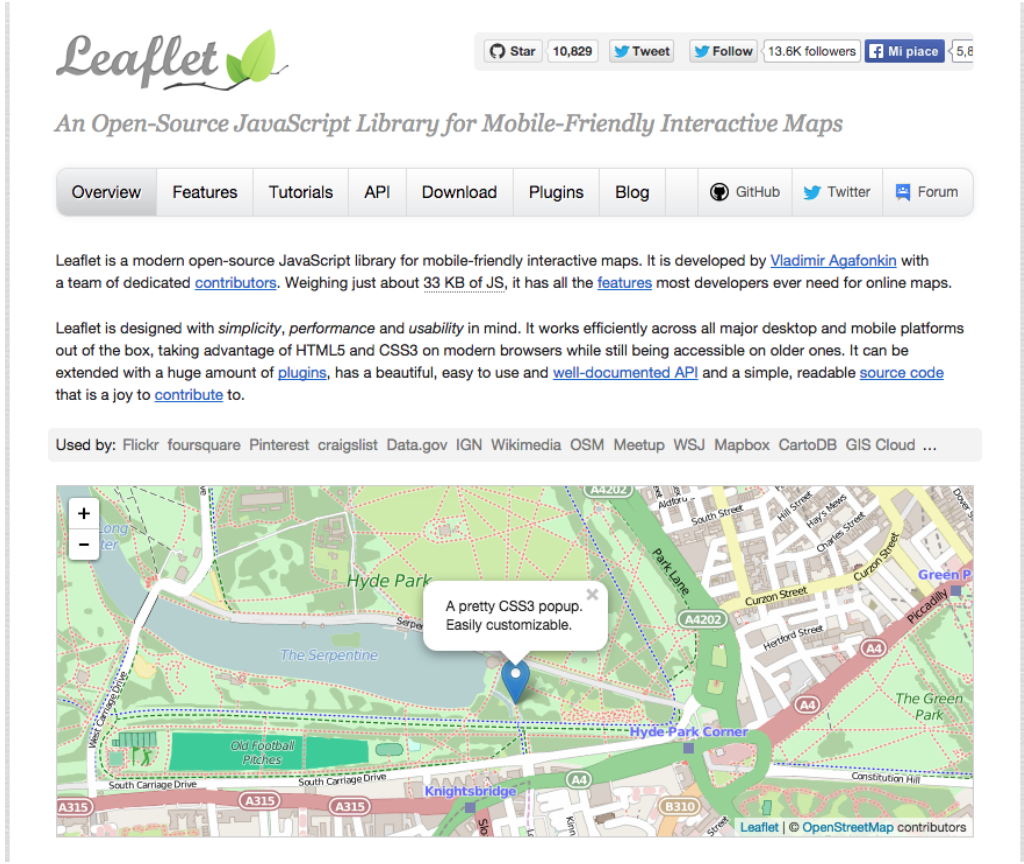
TILE VIEWER EXAMPLE



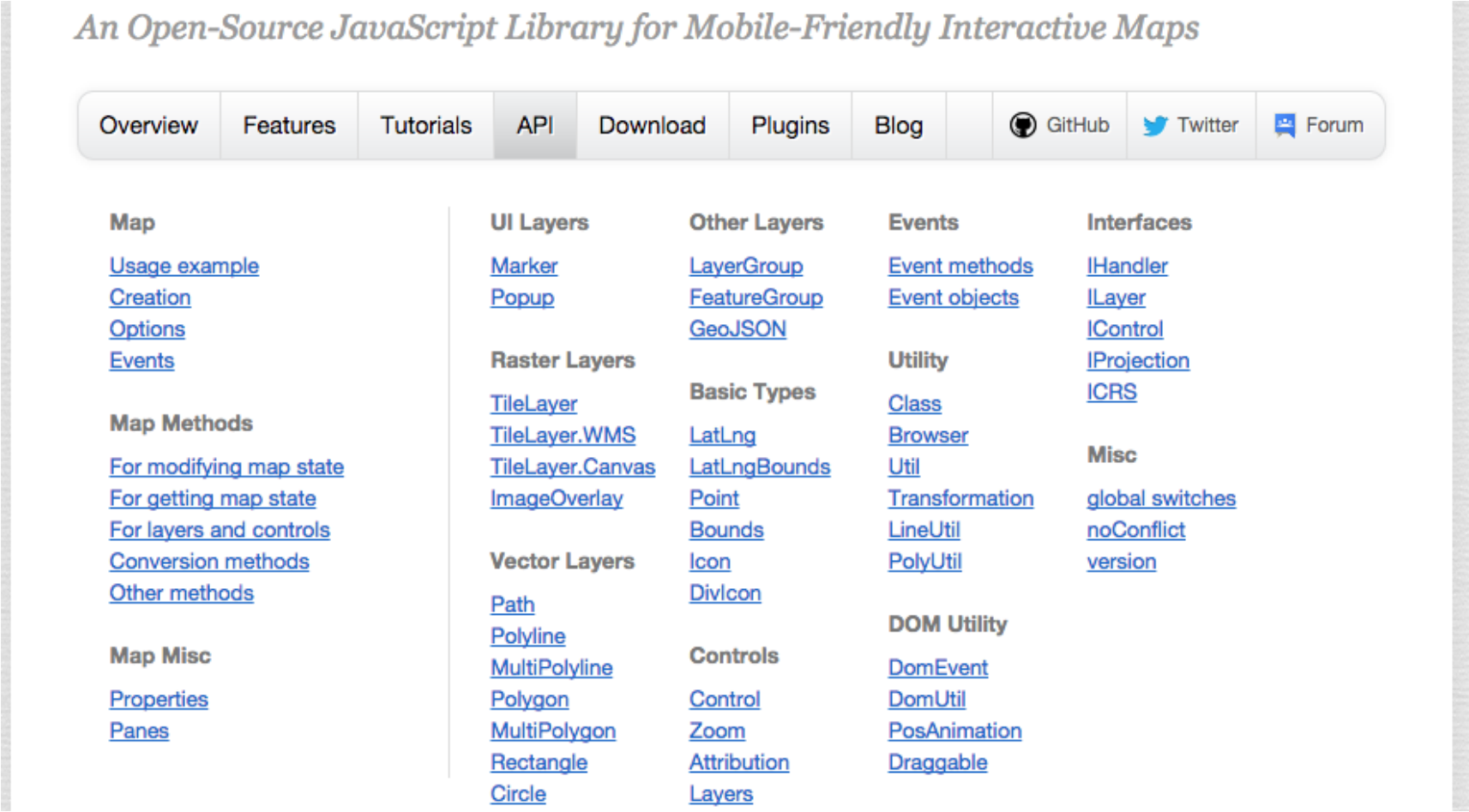
https://bl.ocks.org/mbostock/f9f91fd9148bdc5aa6db

TILE SERVICES

LEAFLET.JS


Leaflet
Star 10,829 Tweet Follow 13.6K followers Mi piace
An Open-Source JavaScript Library for Mobile-Friendly Interactive Maps
Overview Features Tutorials API Download Plugins Blog GitHub Twitter Forum
Leaflet is a modern open-source JavaScript library for mobile-friendly interactive maps. It is developed by Vladimir Agafonkin with a team of dedicated contributors. Weighing just about 33 KB of JS, it has all the features most developers ever need for online maps.
Leaflet is designed with simplicity, performance and usability in mind. It works efficiently across all major desktop and mobile platforms out of the box, taking advantage of HTML5 and CSS3 on modern browsers while still being accessible on older ones. It can be extended with a huge amount of plugins, has a beautiful, easy to use and well-documented API and a simple, readable source code that is a joy to contribute to.
Used by: Flickr foursquare Pinterest craigslist Data.gov IGN Wikimedia OSM Meetup WSJ Mapbox CartoDB GIS Cloud ...
A pretty CSS3 popup.
Easily customizable.
Hyde Park
The Serpentine
Hyde Park Corner
Knightsbridge
The Green Park
Old Football Pitches
Leaflet | © OpenStreetMap contributors

LEAFLET.JS - APIS

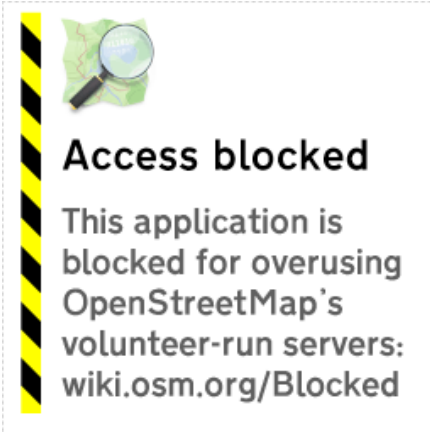


LEAFLET.JS

- A valid tool to provide tile-based maps
  - Open Source
  - Open Data (http://tools.geofabrik.de/mc/)
  - Free
- Easy to use API
- Lightweight lib (only 64k)
- Support mobile applications
- Alternative to Google Maps (http://flink.com.au/tips-tricks/27-reasons-not-to-use-google-maps)

FREE TILES PROVIDERS

- OpenStreetMap
  - Some issues for high traffic services
- MapQuest Open License
  - Free, by attribution
  - Special configuration for heavy usage
- MapBox
  - Free tier
  - Customizable design (see next slide)
  - Same family as Leaflet.js



COMMERCIAL TILE PROVIDERS

- CloudMade
  - Mirror of OSM data till few years ago
  - Leaflet was born here
  - $30 per 1M tiles
- MapBox
  - Free for low traffic
  - $30 for 900k tiles

EASY TO INSTALL/USE

- HTML (Setting the stage)
  - Link CSS (via CDN)
    - <link rel="stylesheet" href="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.css" />
  - Link JS (via CDN)
    - <script src="http://cdnjs.cloudflare.com/ajax/libs/leaflet/0.7.3/leaflet.js"></script>
  - Create a div to contain the map
    -
  - Set height for the container
    - #map { height: 180px; }

EASY TO INSTALL/USE

- Create an object to handle the map
  - var map = L.map('map').setView([51.505, -0.09], 13);

TILE MAP PROVIDERS

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

http://maps.stamen.com

MARKERS AND GEOMETRIES

- var marker = L.marker([51.5, -0.09]).addTo(map);
- var circle = L.circle([51.508, -0.11], 500, {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5
  }).addTo(map);
- var polygon = L.polygon([
    [51.509, -0.08],
    [51.503, -0.06],
    [51.51, -0.047]
  ]).addTo(map);

INTERACTIONS

- marker.bindPopup("Hello world!
I am a popup.").openPopup();
- circle.bindPopup("I am a circle.");
- polygon.bindPopup("I am a polygon.");

EVENT HANDLING

- function onMapClick(e) {
  alert("You clicked the map at " + e.latlng);
  }
- map.on('click', onMapClick);

- var popup = L.popup();
- function onMapClick(e) {
  popup
  .setLatLng(e.latlng)
  .setContent("You clicked the map at " + e.latlng.toString())
  .openOn(map);
  }
- map.on('click', onMapClick);

OTHER EXAMPLES

- Mobile app
  - http://leafletjs.com/examples/mobile.html
- GeoJSON
  - http://leafletjs.com/examples/geojson.html
  - http://geojson.io/
- Tutorials
  - http://leafletjs.com/examples.html