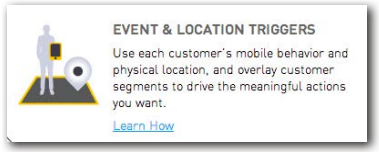
WebMapping: Base concepts

Christian Kaiser
Géovisualisation et traitement de l'information

Warning!

- The content of some slides is already outdated.
- Please beware of updates!
- Web mapping is a huge market.
- Location Based Services even more.
- And it's growing fast.
- Especially Location Based Advertisement…

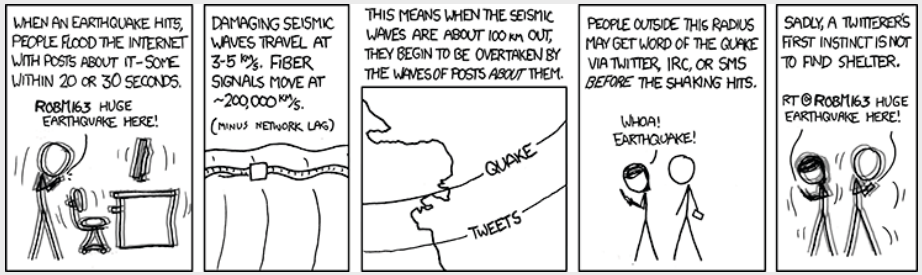


Source: xtify.com (acquired by IBM on 3 October 2013)

The good news…

- Technology changes
- Principles don't change

Why WebMapping?

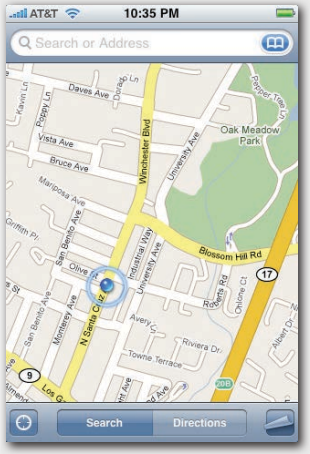


http://xkcd.com/723/

Why WebMapping?

- Great for easily accessing geospatial data from everywhere
- Great to interact with geospatial data
- Great for continuously changing data (easy to update)
- Great for community data, easy to share
- Easy to deploy on many different platforms (desktop computers, mobile devices, microwave, etc…)

WebMapping is changing…



http://www.lukew.com/ff/entry.asp?824

http://www.businessweek.com/news/2012-09-20/apple-maps-lose-way-with-iphone-app-victim-to-google-feud

People are using maps and geospatial data everywhere and at any time!

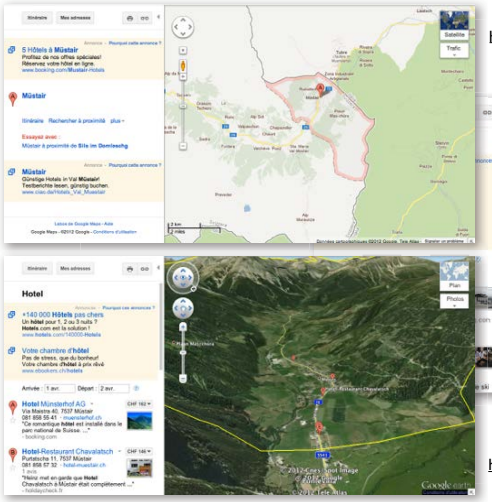
WebMapping

- Access to maps over the Web
- Distribution is cheaper,
  content is more up to date
- But: life expectancy of maps on the Web is shorter («trashable maps»)

WebMapping

- Static maps: map as an image
- Interactive mapping: from view to manipulate
- Dynamic mapping: generated dynamically
- Webmapping and WebGIS

WebMapping = Google

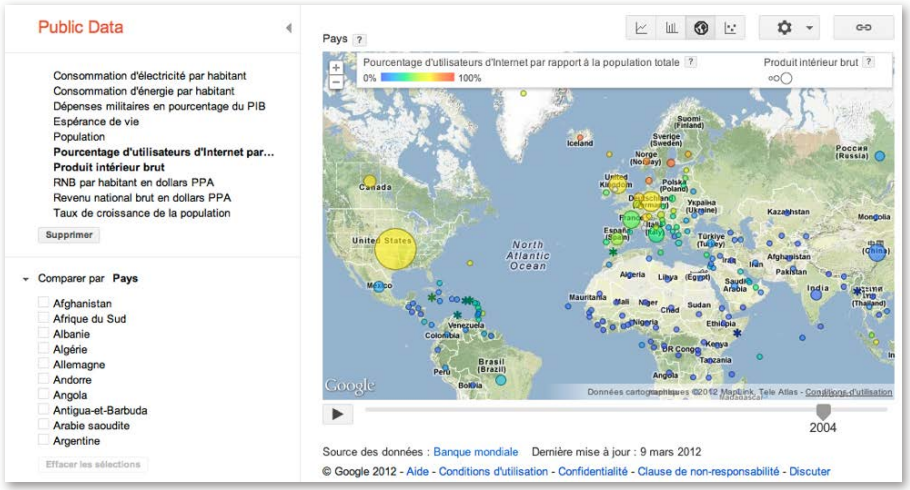


http://g.co/maps/96f6j

http://g.co/maps/4hazp

http://g.co/maps/zcktv

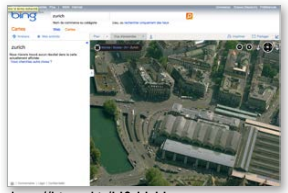
WebMapping = Google



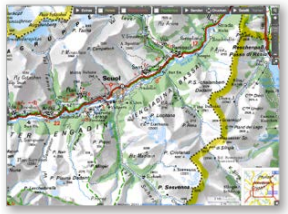
Webmapping > Google !

http://map.geo.admin.ch/

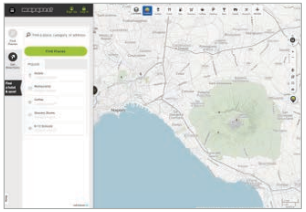
http://here.com



http://binged.it/H0tUsH



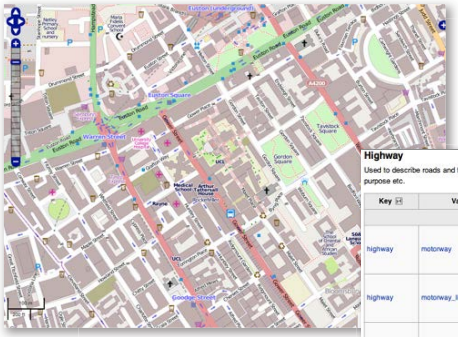
http://de.viamichelin.ch



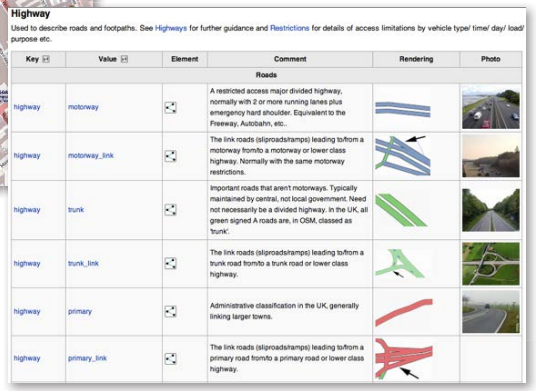
https://www.mapquest.com

etc…

Open Street Map (OSM)

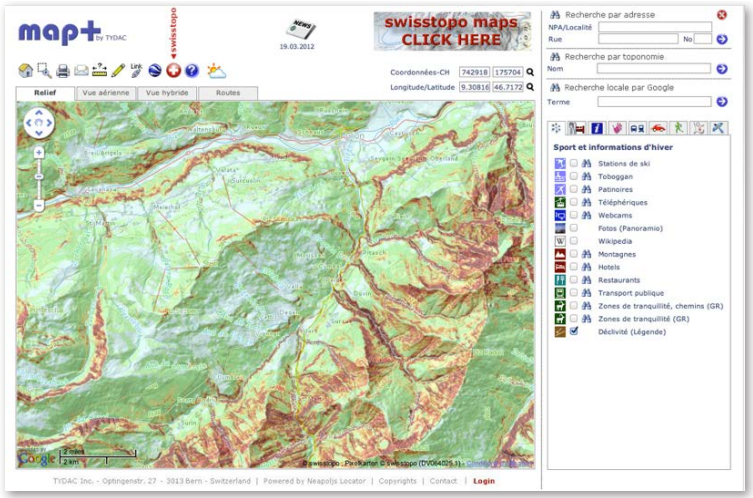


http://www.openstreetmap.org/?lat=51.5240442752838&lon=-0.13325214385986 3&zoom=16



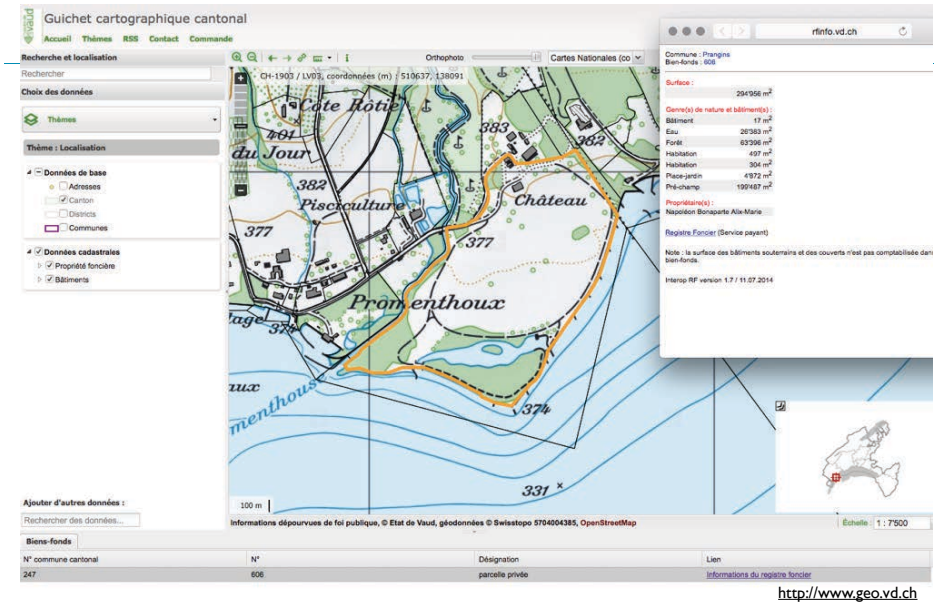
http://wiki.openstreetmap.org/wiki/Map_Features#Highway

Webmapping or WebGIS?



http://www.mapplus.ch?x=732212&y=175675&zl=12

Geoportal

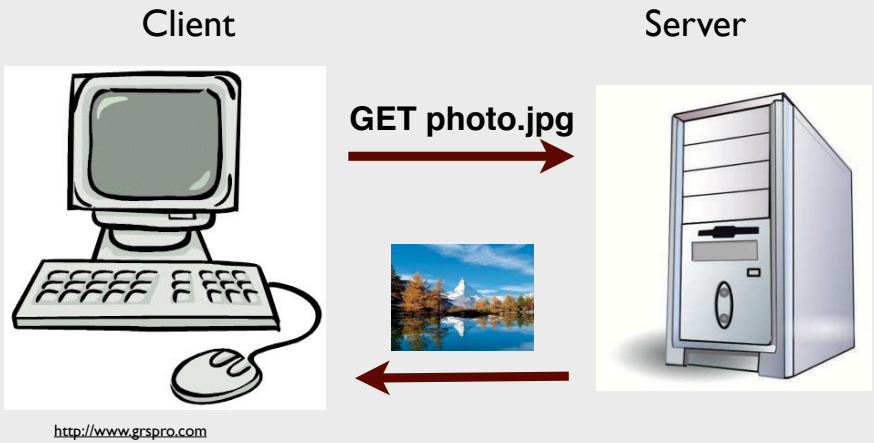


http://www.geo.vd.ch

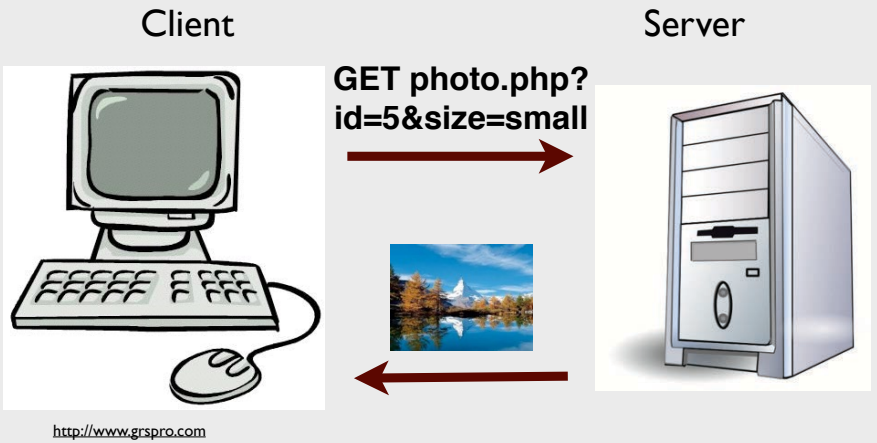
WebMapping: how it works…

- Client / server infrastructure
- Client: Web browser
  - Examples: Firefox, Internet Explorer, Chrome…
- Server: software sending files over HTTP.
  - Examples: Apache, Nginx
  - HTTP = common language (protocol)
  - Client sends file requests to server over HTTP
  - Server returns requested file

WebMapping: how it works…



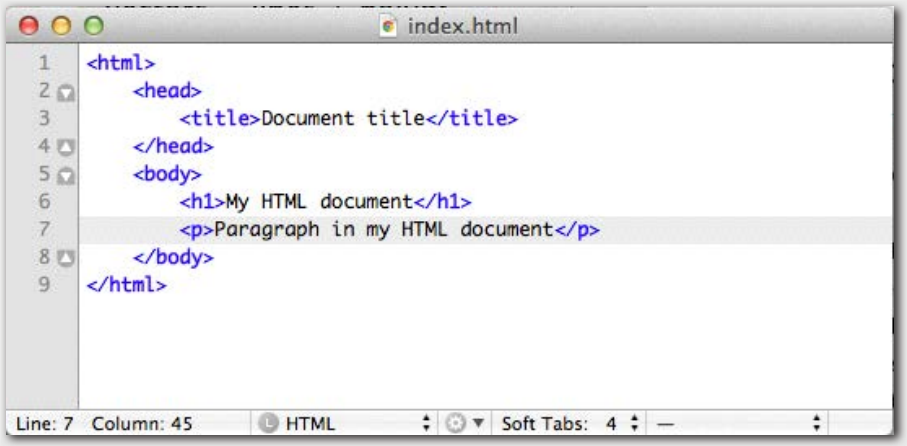
WebMapping: how it works…



Client: the browser

- A Web browser can handle some defined data formats:
  - HTML (HTML 4, hopefully HTML 5)
  - CSS (style sheets)
  - Images: PNG, GIF, JPEG
  - SVG except for some dinosaur browsers (e.g. Internet Explorer 8)
  - sometimes WebGL (3D animations)
  - JavaScript (for manipulation of HTML, CSS, SVG)
  - Sometimes plug-ins: Flash, Silverlight, PDF, Java…

HTML: language of the browser



Maps in the browser

- Ways to bring a map into a Web browser:
  - As an image (PNG, JPEG, GIF)
  - As vector graphics (SVG) for decent browsers
  - As Flash if the plug-in is installed and supported
  - As WebGL
  - As HTML5 on a Canvas element (with Javascript)
- Use Javascript for more interaction and fun!

Image in the browser

- A simple static image in the browser
  - Example: www.swissgeo.ch

Image in the browser: be smart!

- Many static images in the browser
  - Example: maps.google.com
- How to be smart?
  1. Divide your map into many small images (tiles)
  2. Place your images side-by-side
  3. Write some Javascript to move all images around, and to load the tiles that we currently need

Build Google Maps in 6 steps

1. Map tiles: typically 256x256 pixels (regular grid)
2. Limit the number of zoom levels: ~20–25
3. For each zoom level, we produce the whole set of map tiles in advance!
4. We put the images in folders on the server:
   http://my.map.com/X/Y/Z.png
   with X=grid column, Y=grid row, Z=zoom level
5. Add some Javascript for pan & zoom
6. Call the marketing department!

Vectors in a browser

- SVG = Scalable Vector Graphics
- Vectors = lines + polygons: discrete objects
- Geometries are defined by coordinates
- SVG is built on top of XML format:
  stand-alone or embedded in HTML
- SVG is an official Web standard
- Supported by all modern Web browsers

SVG: how it looks…

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg xmlns="http://www.w3.org/2000/svg" version="1.1"
     width="200px" height="300px">

   <circle cx="100" cy="150" r="40" stroke="black"
           stroke-width="2" fill="red" />

</svg>
```

SVG: how it looks…

```
<html>
<body>

<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
   <circle cx="100" cy="150" r="40" stroke="black"
           stroke-width="2" fill="red" />
</svg>

</body>
</html>
```

SVG: who uses it?

- Nobody?
- Everybody!
  - Mostly through a Javascript library
    (with fallback for IE < 9)
  - Google Maps → Get directions
  - OpenLayers → Draw a polygon or vector
- Use of SVG directly possible also
  - Mostly through d3 (d3js.org)

SVG: what else?

- SVG supports animation also
- Possible to embed images
- SVG elements can be manipulated and created through JavaScript, just like standard HTML

Flash: vectors in a plug-in

- Flash can do everything SVG can
- But has video too
- Scripting through own language (ActionScript)
- Creation through GUI: Adobe Flash (.fla)
- Distribution in compiled & compressed format (.swf)

Flash versus SVG

- Advantages of Flash:
  - ⊕ Compact file format, includes everything
  - ⊕ No need to distribute source code
  - ⊕ Many people used to have Flash, it works on most browsers
- Disadvantages of Flash:
  - ⊖ Flash is deprecated
  - ⊖ Proprietary editor needed (Adobe Flash), no standard
  - ⊖ Flash Player not available on all platforms (e.g. iOS, Android)
  - ⊖ No interaction possible with other elements in page

Maps in the browser

- Ways to bring a map into a Web browser:
  - As an image (PNG, JPEG, GIF)
  - As vector graphics (SVG) for recent browsers
  - As Flash if the plug-in is installed and supported
  - As Vector Tiles, together with
    - WebGL
    - HTML5 on a Canvas element (using Javascript)
- Use Javascript for more interaction and fun!

What is the easy way to get a map in a browser?

- No free lunch!
  - Use a GUI program to prepare your map
    e.g. Adobe Flash, Inkscape (SVG), …
  - Use existing libraries
    e.g. Google Maps, Leaflet, Swisstopo API, OpenLayers, d3, …
  - Using an online service designed for mapping
    e.g. Mapbox, ArcGIS Online, …
- But: we still need to prepare the data
- Often some more advanced expertise needed (JS)

Using a Javascript library

- A library is a collection of functions and objects that we can use through a well-defined and hopefully documented API (Application Programming Interface)
- The data formats depend on the library
  → we need to adapt ourselves!

Using a JS mapping library

- Raster data:
  - XYZ tiles
  - WMS (Web Mapping Service)
  - WMTS (Web Mapping Tile Service)
- Vector data (geometry + attributes):
  - KML / KMZ
  - GeoJSON
  - WFS (Web Feature Service)
  - Mapbox Vector Tiles

GeoJSON format

- Format for geometries + attributes
  - Extension of JSON (JavaScript Object Notation), which is used a lot in Web sites with JavaScript
  - It basically contains simple JavaScript objects and variables
- Easy to read in JavaScript
  (because it is JavaScript already)

GeoJSON: a closer look

```
{ "type": "FeatureCollection",
  "features": [
    { "type": "Feature",
      "geometry": {
          "type": "Point",
          "coordinates": [102.0, 0.5]
      },
      "properties": {
          "description": "a simple point",
          "temperature": 10.5,
      }
    }, { ... another Feature ... }, ...
  ]
}
```

GeoJSON format: how to create

- QuantumGIS
- PostGIS (ST_AsGeoJSON)
- ogr2ogr (command line utility part of GDAL)
- Using a script: many libraries around…
- ArcGIS: JSON toolset
- Online service (e.g. http://2geojson.com)

GeoJSON format: plus-minus

- Advantages:
  - ⊕ Widely adopted (e.g. Twitter uses GeoJSON)
  - ⊕ Easy to create programmatically, fast to parse
  - ⊕ Human readable
- Disadvantages:
  - ⊖ Needs additional library for GoogleMaps (e.g. GeoJason)
  - ⊖ No validation for data structure

TopoJSON: the brother of GeoJSON

- TopoJSON is an evolution of GeoJSON
- Smaller files, no redundant lines stored in file
- Supports generalization
- Easy to convert to GeoJSON on client

OGC Web services

- OGC is a consortium defining open geo-spatial data standards (≠ open-source !)
- OGC defines several Web services
  - A Web service allows retrieving data in a precise format through a URL-based API (request → response)
  - OGC Web service responses are XML encoded data
- Generally, we only need to know the base URL of the service to use it

OGC Web services

- WMS = Web Mapping Service: for raster data
- WMTS = Web Mapping Tile Service: for raster data coming in tiles
- WFS = Web Feature Service: for vector data (geometries + attributes)
- Other less common services exist also (see www.opengeospatial.org)

OGC Web services

- We can use Web services also with traditional GIS software
- There are few reliable and fast Web services around
- Swisstopo provides some data as a WMS: search for "Swisstopo WMS" in your favourite search engine (DuckDuckGo)

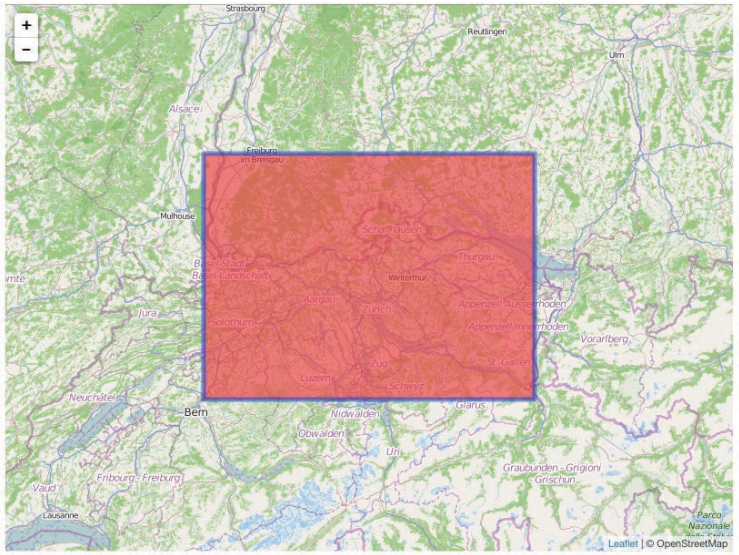
Mapbox Vector Tiles

- Cut vector data in regular tiles
- Efficient encoding for map data
- Rendering on client-side
- Fast!
- But not easy to implement…

Where to start?

- We start with the easy stuff, of course:
  - HTML page
  - Interactive map using Javascript library «Leaflet»
  - Layers to add in GeoJSON format

GeoJSON Example



Short Leaflet example

```
<html>
   <head>
      <link rel="stylesheet" href="leaflet.css" />
      <script src="leaflet.js"></script>
      <script src="script.js"></script>
   </head>
   <body onload="main()">
      <h1 id="title">GeoJSON Example</h1>
      <div id="map" style="background-color: #666;
                    width: 800px; height: 600px;"></div>
   </body>
</html>
```

```
function main() {
   map = L.map("map").setView([8.5, 47.5], 8)
   L.tileLayer("http://{s}.tile.osm.org/{z}/{x}/{y}.png",
      { attribution: "&copy OpenStreetMap" }).addTo(map);

   var fc = {
      "type": "FeatureCollection",
      "features": [{
         "type": "Feature", "geometry": {
            "type": "Polygon",
            "coordinates": [[ [7.5, 47.0], [9.5, 47.0],
                  [9.5, 48.0], [7.5, 48.0], [7.5, 47.0] ]]
         }, "properties": { "fid": 1 } }]
   };
   L.geoJson(fc.features, {
      style: {fillColor: "#f33", fillOpacity: 0.6}
   }).addTo(map);
}
```