International Cartographic Association
Association Cartographique Internationale

Automatic Labels in Leaflet

Mátyás Gede[a, *]

[a] *Institute of Cartography and Geoinformatics, ELTE Eötvös Loránd University, saman@inf.elte.hu*

* Corresponding author

Abstract: Automatic labelling is a lacking feature in the popular client-side web mapping library, Leaflet. Although there exist various workarounds and substitutes, most of these do not fulfil the basic cartographic rules, e.g. avoiding text overlaps.

This paper discusses the various workarounds addressing this problem, and introduces the author's solution, which extends Leaflet's GeoJSON class with automatic labelling functionality. The new class displays labels for point, line or polygon features. Labels can be placed either on right or left side of points or automatically for better fitting. Markers for points are only displayed if their labels can be drawn without overlaps. Label text is specified either by property field name or a user-specified function. It is also possible to specify priority order of features (higher priority labels are drawn first) and to customize label styles.

Keywords: automatic labels, Leaflet, webcartography, open source

1. Introduction

Labels are essential part of maps. As this area was well researched in the past decades, most desktop GIS software has sophisticated solutions for map labelling (Brewer and Frye, 2005). Server-side web mapping software such as MapServer (OSGeo 2022) or GeoServer (GeoServer 2022) also provide dynamic labelling tools. These can even be controlled from client side using Styled Layer Descriptors (SLD) in the map requests (Lupp 2007).

Compared to desktop and server-side environment, labels are treated as stepchildren in client-side web mapping. Most JavaScript libraries, especially open source ones provide no or limited support for automatic labelling of features (Brinkhoff 2017). This leads to poorly designed web maps flooding the Internet – people want to share information using maps but don't want to spend too much time with that, so they just use the built-in features of libraries, even if those are cartographically inappropriate.

In traditional cartography there are strict rules of positioning names on maps. According to Imhof (1975) the general principles are:

- legibility: the names should be easily read, discriminated, located,
- clear graphic association: the name and the object to which belongs should be easily recognised,
- names should disturb other map content as little as possible,
- names should assist directly spatial situation, connections, etc.,
- type arrangement should reflect the classification and hierarchy of objects,
- names should not be evenly dispersed over the map, nor should names be densely clustered.

Considering all these principles meant tremendous manual work on name placing. With the emerge of automatically created maps these concepts first took a back seat. Later, most of them were incorporated into various GIS systems, but as the algorithms in the background of a decent label placing system are rather complex (Doddi et al, 1997), they didn't really appear in client-side web mapping.

Naturally, not all the traditional principles can, or should be implemented in the case of dynamic, zoomable web maps. The possibility free zooming and panning of the map allows more slack labelling as increasing the zoom gives more space to display more names. At the same time, it raises new challenges: labels on lower zoom levels should be selected using the rules of cartographic generalization.

This paper introduces a possible solution created for the popular open-source web mapping framework Leaflet, addressing at least a subset of Imhof's name placing principles.

1.1 Similar work

Naturally, there are other research projects on this topic. Brinkhoff (2017) – besides giving a thorough overview on the subject and also suggesting extensions on standards such as Symbology Encoding (Müller 2006), – introduces a prototype solution to be used with Google Maps JavaScript API. His solution is reported to work fast even with several thousand points but only deals with point objects.

Kenta Hakoishi's *Leaflet.LabelTextCollision* is a plugin implementing a Canvas renderer extension that displays labels and detects collisions (Hakoishi 2016). Unfortunately this plugin has not been updated for six years now, and lacks formatting options such as label

alignment, font settings and other styling possibilities; and there is no possibility to handle icons and labels together.

A further extension from 3Maps, *Leaflet.streetlabels* (Santos & Dias 2022) combines Hakoishi's work with *Canvas-TextPath* (Viglino 2016) to support labels along polylines. This plugin also allows a limited text style customisation: the font size and colour, and the text halo properties can be set.

Other solutions simply use Leaflet's tooltip object or markers with a custom defined HTML element instead icon which can be a workaround for some cases but does not solve most problems, especially text collision.

2. Labelling features of popular open-source client-side web mapping libraries

Farkas (2017) thoroughly examined the various web mapping libraries. Based on his work, the most usable client-side libraries are OpenLayers and Leaflet. There are numerous other libraries as well but they are either not totally open (e.g. the new version of MapBox GL JS requires an access token even for instantiating the Map object) (Mapbox 2022) or are less known or have very limited cartographic capabilities.

2.1 OpenLayers

According to Farkas (2017), OpenLayers is the most comprehensive client-side web mapping library available. Vector features can be labelled as a part of their styling. Label formatting options are rich and (just like any other styling) may depend on feature attributes, which makes it possible to differentiate various feature classes by their label style (OpenLayers 2022). Label text can be rotated or fitted to lines as well. By default, line and polygon labels are only drawn on a specific zoom level if they fit into the corresponding feature. This setting also prevent label placing conflicts. Using the `declutter` option on a vector layer, there is also conflict solving for point features and their labels, which, together with setting `renderOrder` (a function that takes two features as arguments and the sign of the return value is used to determine the drawing order of features) offers a great solution for prioritised labelling. There is one issue here: the current viewbox of the map is not considered when rendering labels, so some names may partially fall outside the view (Figure 1).

A big disadvantage of OpenLayers is its harder learning curve. Although its features are much richer than the other popular library, Leaflet, non-expert users (especially ones with limited previous programming skills) prefer this latter one because it is much easier to get started with.

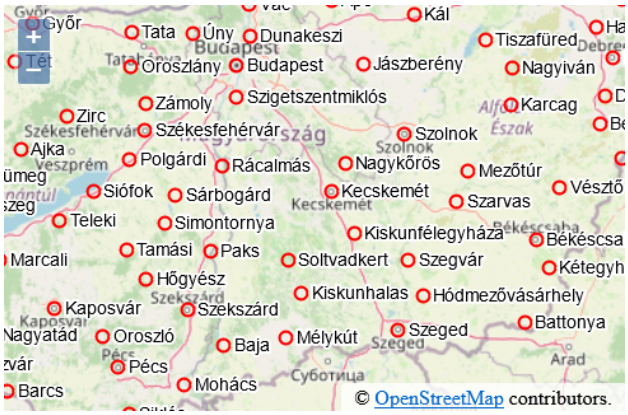

Figure 1. Decluttered labelling in OpenLayers.

2.2 Leaflet

Despite its limitations when compared to OpenLayers, Leaflet is also very popular among web developers. Its biggest advantage is simplicity: the most often needed functions of an interactive web map can be implemented with a few simple lines of code.

Leaflet has no built-in labelling solution. There are, however, various plugins and workarounds to display names on the map. One possible way is to create markers without an icon, but with a custom HTML content, using the DivIcon class (Figure 2). The disadvantage of this solution is that it creates a label that is an independent map feature, not connecting to the map symbol the name belongs to.

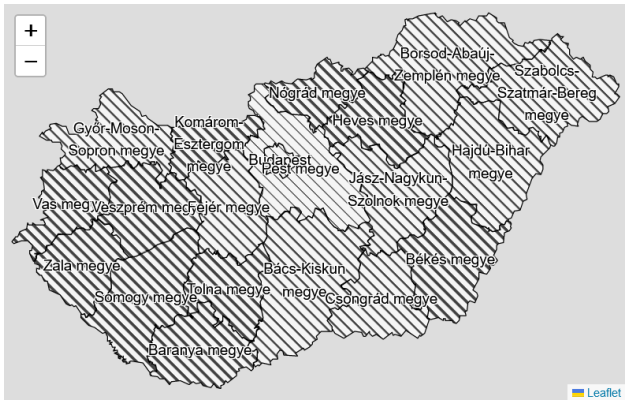

Figure 2. Labelled polygons using L.DivIcon

Another solution is the use of Leaflet's tooltips with the `permanent` option set to true. Tooltips were originally designed to appear only when user hovers the mouse pointer over a feature, but with this workaround they will be always visible (Figure 3). On the other hand, one needs a lot of extra CSS rules to get rid of the default "bubble" encapsulating the tooltip text.

Figure 3: Labels implemented as permanent tooltips

None of the workarounds above can do anything with label collision conflicts, nor any other of Imhof's principles.

2.3 Dynamic labelling features web mapping libraries should provide

In order to help creating easily usable, informative maps, a web mapping library should offer the followings:

- The possibility of adding dynamic labels to features. Label text and also its styling might depend on attributes.

- These labels should not overlap with each other or with any point symbols. Web maps are dynamically zoomable, therefore no need to display all names all time, only that much that can be fitted into the current view.

- Some point feature classes – especially settlements – generally only appear together, i.e. if a label is not displayed because it cannot fit without overlaps, the corresponding point symbol should also be removed.

- If not all labels are displayed all time, there should be a possibility to set a priority order.

3. Features of 'leaflet-labeler'

The author developed a subclass of Leaflet's GeoJSON layer. This class (the work name is Labeler) displays labels upon features based on either an attribute or a user defined function. The labels are dynamically placed, to avoid conflicts. Point symbols are only displayed if their symbols as well as the corresponding label can be displayed without overlaps.

3.1 Label placing mechanism

In order to save computing time, label placing mechanism is rather simple: when creating the layer, features are stored sorted based on their priority. On every update of the map (change of the viewport bounding box due to map move or zoom) bounding boxes of labels (in pixel coordinates) are calculated before displaying, and they are only placed to the map if the bounding box fits the current view and has no conflict with the bounding boxes of the labels already displayed. Point symbols are treated the same, their bounding boxes are also taken into account. Currently there are four possible positioning settings for labels: centered (suggested for lines and polygons), left and right (for points), and automatic, which means left and right positions are also tested.

Labels are HTML objects, having a specific class name, therefore it is easy to apply various styles on them. As CSS currently does not support "halo" effect for texts, (but an outline is usually important for labels on web maps) it is implemented by the `text-shadow` CSS property.

3.2 Usage of the Labeler class

The main goal was to provide a simple solution for the most typical needs. Therefore, after including the JavaScript and the CSS file of *leaflet-labeler* in the code, adding dynamic labels to a GeoJSON layer might be as simple as changing `L.GeoJSON` to `L.Labeler` in the JavaScript code, in case feature names are stored in the `name` attribute and the default label style fits the mapmaker's needs.

There are a bunch of options that can be used to customize labels.

- It is possible to specifiy a property that stores label text. Alternatively, a labelling function also can be defined (in this case the layer object of each feature is passed as parameter to the function – just like in the case of Leaflet's built-in `bindPopup` and `bindTooltip` functions).
- Default label style can be overridden either by an object literal or a function returning an object of CSS settings.
- Point labels are displayed together with the corresponding marker. If the label is not placeable without conflict, the marker is also hidden.
- It is possible to apply a filter on features not only on layer creation (which is already a feature of L.GeoJSON) but at any time. This is useful when one wants to dynamically set which subset of features should be displayed on the map.

The extension (with examples) is available on Github, at *https://github.com/samanbey/leaflet-labeler*.

3.3 Examples

Figure 4 shows a point feature layer before and after zooming in. It also demonstrates the use of a labelling and a styling function (labels are composed of settlement names and their population; major cities are written in upper case and bold letters).

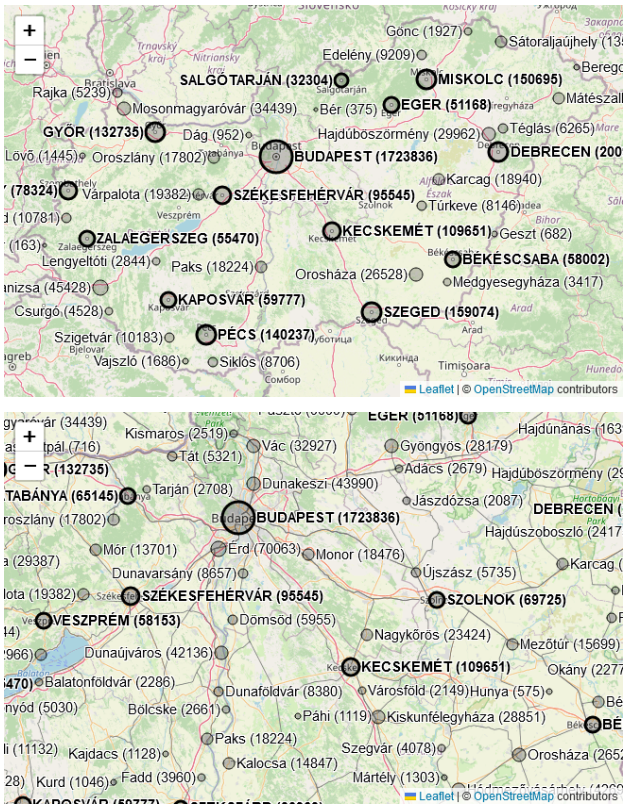

Figure 4. Points with labels – the same map with different zoom settings.

Figure 5 displays a map with polygons labelled. This example also features hatch fill, implemented by *leaflet-hatchclass* (Gede 2022). Although the default label style settings include non-wrapping whitespaces (forcing single line labels), it is overridden here to display long county names in multiple lines if necessary.

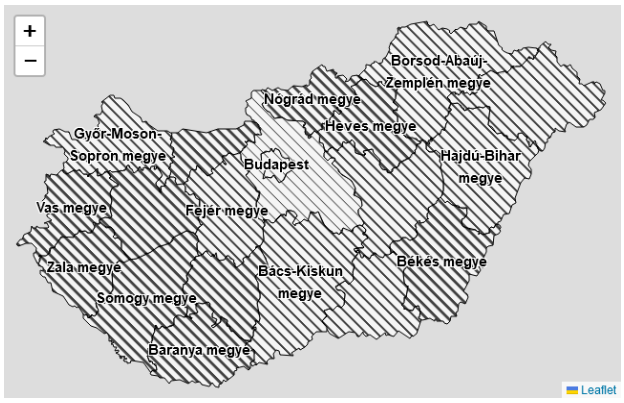

Figure 5. Polygons with labels.

Figure 6 shows lines with labels (a road network with road numbers). Displaying curved labels along lines (e.g. for street names) is not supported yet. It is possible, however, to put a "box" around labels, using simple CSS rules. Taking advantage of the possibility of text styling functions, motorways are differentiated on the map by blue background.

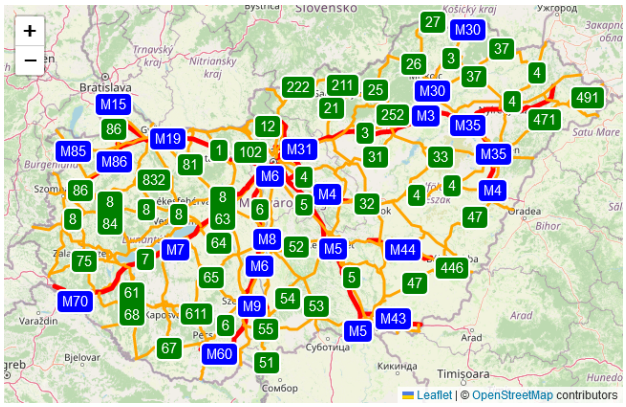

Figure 6. Lines with labels.

3.4 Performance

Rendering speed was tested on a notebook with Core i7 CPU. The test dataset contains settlements of Hungary as points (around 3200 objects). Loading or updating the map after viewbox change takes typically 0.4–0.7 seconds, regardless the browser used (tested on Google Chrome, Mozilla Firefox and Microsoft Edge). With only 500 objects, rendering is under 0.1 seconds. These delays mean that visualisation of large datasets is still enjoyable.

3.5 Conclusions, future plans

Recognising the need of an easy-to-use solution for automatic labelling in client-side web maps, the author developed a tool that extends Leaflet to display labels for point, polygon or line objects. Labels are highly customizable, their text and style can be set based on feature properties.

This is an ongoing project; there are some limitations as well: label placing conflicts are detected only within the layer. (If there are more layers with labels, conflicts between texts belonging to different layers are not resolved.) Labels for lines can only be placed to the centroid of the line (no rotated or curved labels).

Future plans include the possibility of fitting labels on curves (for example street or river names or labelled contour lines), an option to force polygon labels inside the corresponding polygons, and to make multiple instances of layers with labels detect label placing conflicts together. This latter might be realized by extending a renderer class instead of a layer class, which would also make it possible to use data sources other than GeoJSON for labelling.

4. References

Agafonkin, V., 2022. Leaflet API reference. https://leafletjs.com/reference.html

Brewer, C.A., and Frye, C., 2005. Comparison of GIS and Graphics Software for Advanced Cartographic Symbolization and Labeling: Five GIS Projects. Proceedings of the 22nd ICC, A Coruña, Spain, 2005.

Brinkhoff, T., 2017. Supporting Dynamic Labeling in Web Map Applications. https://agile-

online.org/images/conferences/2017/documents/shortpapers/80_ShortPaper_in_PDF.pdf

Doddi, S., Marathe, M.V., Mirzaian, A., Moret, B.M.E., Zhu, B., 1997. Map labeling and its generalizations. Proceedings of the 8th SIAM Symposium on Discrete Algorithms SODA'97, pp. 148–157.

Farkas, G., 2017. Applicability of open-source web mapping libraries for building massive Web GIS clients. In: *Journal of Geographical Systems*, Springer, 19(4), pp. 273–295.

Gede, M., 2022. Hatch Fill on Webmaps – to Do or Not to Do, and How to Do, Abstr. Int. Cartogr. Assoc., 5, 48, https://doi.org/10.5194/ica-abs-5-48-2022

GeoServer, 2022. GeoServer documentation. https://docs.geoserver.org/

Hakoishi, K., 2016. Leaflet.LabelTextCollision. https://github.com/yakitoritabetai/Leaflet.LabelTextCollision

Imhof, E., 1975. Positioning Names on Maps, The American Cartographer, 2:2, 128-144, DOI: 10.1559/152304075784313304

Lupp, M. (ed.), 2007. OGC Styled Layer Descriptor profile of the Web Map Service Implementation Specification, Version 1.1.0 (revision 4), OGC 05-078r4. https://portal.ogc.org/files/?artifact_id=22364

Mapbox, 2022. Migrate to Mapbox GL JS v2. https://docs.mapbox.com/mapbox-gl-js/guides/migrate-to-v2/

Müller, M., 2006. OGC Symbology Encoding Implementation Specification, Version 1.1.0 (revision 4). OGC 05-077r4

OpenLayers, 2022. OpenLayers Documentation. https://openlayers.org/doc/

OSGeo, 2022. MapServer 8.0.0 Documentation. https://mapserver.org/documentation.html

Santos, J., Dias, L., 2022. Leaflet.streetlabels. https://github.com/3mapslab/Leaflet.streetlabels

Viglino, J-M., 2016. Canvas-TextPath. https://github.com/Viglino/Canvas-TextPath