International Cartographic Association
Association Cartographique Internationale

Auto-generalized labels on multi-layer Leaflet maps

Mátyás Gede [a,*]

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

* Corresponding author

Abstract: Leaflet is one of the most popular client-side web mapping libraries. It is lightweight, easy-to-use, especially for ones without strong programming background. The library, however, lacks a very important feature: map labels.

The author developed a plugin for Leaflet that makes it easy to show map labels for any vector layer. Labels are automatically generalized to avoid overlapping texts. Point symbols or markers can be linked to their labels i.e. if the label cannot be displayed, its marker is also removed. Labels are drawn in priority order. Priorities, as well as label texts and styles can be highly customized with respect to feature/layer properties.

Labels are displayed as HTML <*span*> elements, allowing developers to create complex labels with various borders or backgrounds as well, by CSS styling rules.

The plugin extends the Map class, while label properties can be set for each layer separately. Collision conflicts are checked between layers as well. Dynamic addition/removal of map objects is also supported. Performance of the plugin was tested with 3700 map objects (mixed data of points, polygons, and lines). Updating map labels after zoom/pan required only fragments of seconds.

The source code, user guide and examples are available at https://github.com/samanbey/leaflet-mapwithlabels

Keywords: automatic labels; Leaflet; web cartography; 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.

Proceedings of the International Cartographic Association, 7, 2, 2025 | https://doi.org/10.5194/ica-proc-7-2-2025
32nd International Cartographic Conference (ICC 2025), 17–22 August 2025, Vancouver, Canada.

1.1 Previous Works

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 seven 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 customization: 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 an icon which can be a workaround for some cases but does not solve most problems, especially text collision.

The solution presented in this paper is based on a previous work of the author, the *leaflet-labeler* plugin (Gede, 2023). That plugin implemented labelling on a subclass of Leaflet's *GeoJSON* layer. While it is perfectly useable for maps with only one layer with labels, label collisions between layers are not checked.

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 prioritized labelling. There is one issue here: the current view box of the map is not considered when rendering labels, so some names may partially fall outside the view (Figure 1).

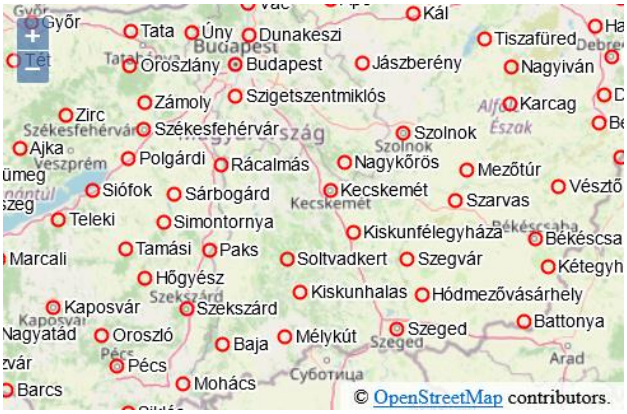

Figure 1. Decluttered labelling in OpenLayers.

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.

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.

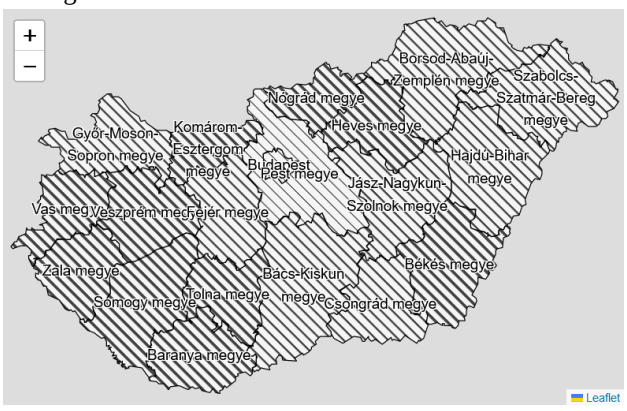

Figure 2. Labelled polygons using L.DivIcon.

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.

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

Figure 3: Labels implemented as permanent tooltips.

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 as well as its styling might depend on feature 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 as many that can be fitted into the current view.
- Some point feature classes – especially settlements – and their labels 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. The leaflet-mapwithlabels Extension

The author developed an extension to Leaflet called *leaflet-mapwithlabels*, which implements a subclass of *L.Map*, the map interface class of Leaflet. Using the new class, the features of any layer that has its `label` option set will be labelled. Labels are dynamically positioned and generalized to avoid overlaps. In the case of point symbols, markers, it is possible to bind the symbol to the label which means that the symbol will only be displayed if its label also fits to the map.

Labelling is highly customizable. Label text, style and priority can be set either as literals or as functions of the corresponding layer object. For linear features, labels can be repeated in specified distances along the line.

3.1 Under the Hood

Label placing mechanism is inherited from the author’s previous project (Gede 2023), but there are some key changes in order to provide cross-layer label collision detection. Labels are stored in a list. When a new layer with label is added or removed, this list is updated. The information stored for each label is label text, reference point coordinates, geometry type, priority and in the case of point geometry, the size and the anchor point of the symbol. When updating labels, this list is iterated over, in the order defined by priority. A label is displayed only if it has no conflict with already displayed (i.e. higher priority) labels. In the case of point objects, not only text collision checked but symbol collision as well. Conflicts are tested by checking the intersection of the text/symbol bounding boxes. For linear features it is possible to repeat labels along the line in specified pixel distances. In this case each instance of the repeated labels is tested for collision.

Labels are HTML objects, having a specific class name (leaflet-label), 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 in the default style.

3.2 Using the extension

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-mapwithlabels* in the code, and changing `L.Map` to `L.MapWithLabels`, map objects of any layer can be labelled by setting the corresponding layer’s `label` option, either as a string literal or as a function assigning text to layer objects (similarly to the way of binding tooltip or popup texts to feature group layers).

The style and the behaviour of labels can be customized on layer level by a handful of additional options: the gap between the symbol and the label, priority, label position, as well as the style of the label object.

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 5 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.

Proceedings of the International Cartographic Association, 7, 2, 2025 | https://doi.org/10.5194/ica-proc-7-2-2025
32nd International Cartographic Conference (ICC 2025), 17–22 August 2025, Vancouver, Canada.

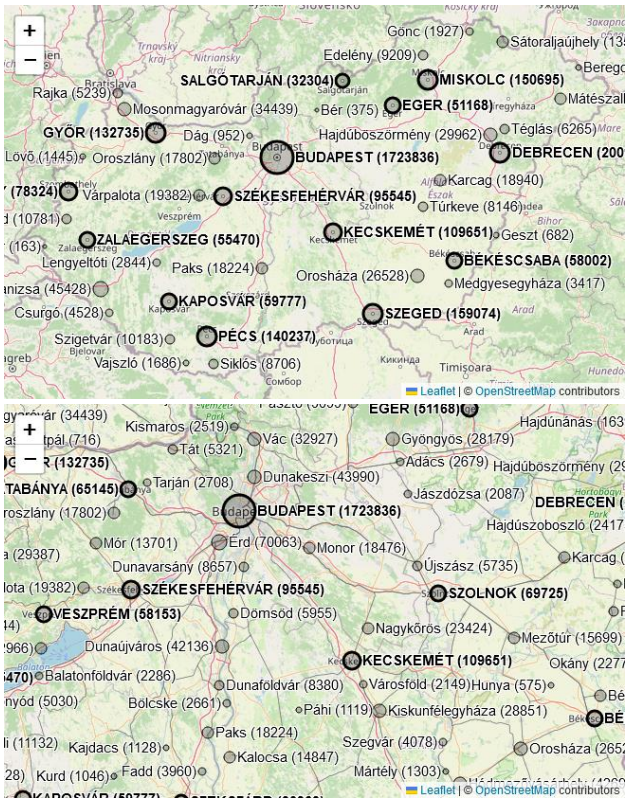

Figure 4. Points with labels – the same map with different zoom settings.

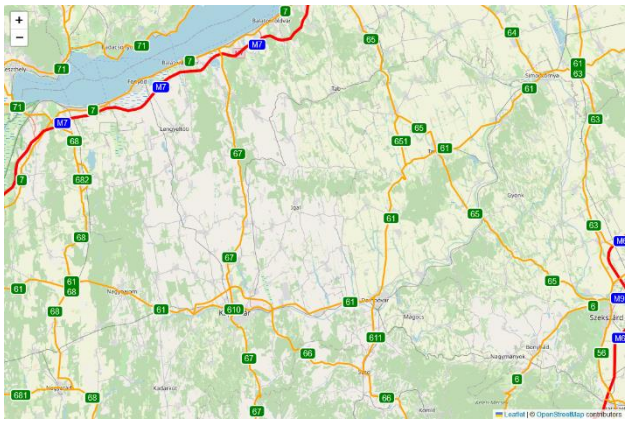

Figure 5. Lines with labels.

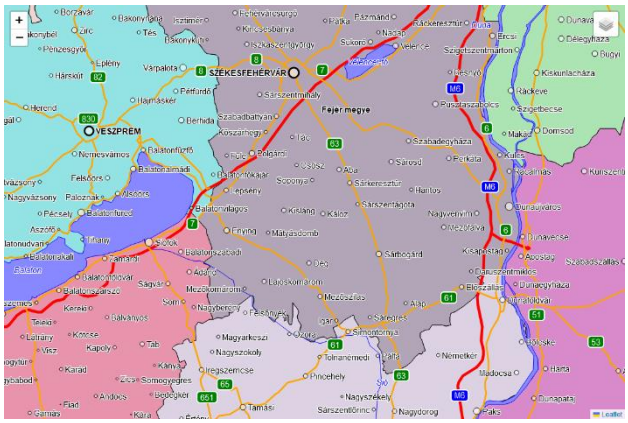

Figure 6. Several layers with labels on the same map.

Figure 6 shows a map with several layers to demonstrate cross-layer label collision check.

3.4 Performance

Rendering speed was tested on a notebook with Core i7 CPU. The test dataset contains settlements of Hungary as points, supplemented by hydrography, administrative areas and major road network (around 3200 point, 400 polyline, 25 polygon 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 Known issues

L.GeoJSON layer does not pass over options to its custom Marker layers, therefore if custom markers are used for GeoJSON points, label-specific options have to be included within marker factory function options. Example code is included on the project's GitHub page.

4. Conclusions, future plans

Recognizing 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 layer or feature properties.

Future plans include the possibility of fitting labels on curves (for example street or river names or labelled contour lines), and an option to force polygon labels inside the corresponding polygons.

5. References

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

Brewer, C.A., & 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. 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

Gede, M. (2023). Automatic Labels in Leaflet. Adv. Cartogr. GIScience Int. Cartogr. Assoc., 4, 8, https://doi.org/10.5194/ica-adv-4-8-2023

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