A Low-Code Approach for Creating Dynamic Map-Based Web Applications Using W3C Web Components

Andreas Schmidt*‡ and Tobias Münch†
* University of Applied Sciences
Karlsruhe, Germany
Email: andreas.schmidt@h-ka.de
‡ Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: andreas.schmidt@kit.edu
† Münch Ges. für IT Solutions mbH, Germany
Email: to.muench@muench-its.de

***Abstract*—We present a set of web components that enable the declarative development of web-based, map-centered applications in a simple way. To do this, we used the World Wide Web Consortium (W3C) standard *Web Components*, which enables the development of new HTML elements. The developed components encapsulate the functionality of the *leaflet library*, a widely used Javascript library for the realization of map-based applications. The declarative approach makes it possible for non-programmers to develop applications within a short time. Some of the developed components have interfaces for accessing server-side information, such as *GeoJSON*-data sources and time series databases. This makes it possible to develop information-rich, "live" applications with our components.**

Keywords-dynamic interactive map applications; low-code; web-components; geojson; time series databases.

I. Introduction

For geospatial data, maps are the ideal form of presentation. The Open Street Map initiative [1] has created a global, freely available dataset that is suitable for displaying electronic maps of any scale. *Leaflet* [2] is a popular JavaScript library for developing web-based map applications that run in the browser.

A. Leaflet

The main visual components of the *leaflet library* are basemaps, overlay-layers, and geometric elements, such as markers, lines, polylines, circles, polygons (closed polylines), and rectangles.

The map displayed is composed of a basemap and optional overlay-layers. There are many freely available basemaps for *leaflet*, many of which are based on the open street map dataset [1], but there are also basemaps from other providers, such as Google, ESRI, etc. Overlays consist of images (i.e., png, svg) that are placed on top of the baselayer and enrich it with additional information, such as nautical marks [3] or hiking routes. Overlays can be shown or hidden on the map as needed.

Markers represent specific points on a map and can be displayed either with a standard visualization or with your own icon. The graphical elements can be enriched with *tooltips* and *popup menus*. In addition, leaflet has a wide range of events to which you can attach your own functions.

Graphic components, such as markers, lines and polygons can be grouped in so-called *layer-groups*. The idea behind this is that these can then be easily shown or hidden with a single instruction, analogous to the overlays.

The GeoJSON class allows the display of graphical features that are described in the form of GeoJSON [4] data sets. These can be points, (multi) lines, (multi) polygons, or geometric collections (multiple instances of the previous types).

B. Web-Components

The leaflet library is one of the most widely used JavaScript libraries for developing map-based applications. Its advantage lies in its technical maturity and good documentation. Nevertheless, the library requires in-depth knowledge of the underlying classes and programming experience with JavaScript to develop even the simplest applications. In contrast, this work takes a low-code approach in which the components of the card-based application are defined declaratively. The W3C composite standard *web-components* [5] is used for this purpose. A web component is a JavaScript class that must implement a series of methods in order to be integrated into the Document Object Model (DOM) tree within a website. The class is then mapped to an HTML tag-name that can then be used within the website.

The main goal of our work is to provide the leaflet library with a new declarative interface based on the W3C standard Web Components, so that non-programmers are also able to create map-based applications. But our components also make things easier for the experienced developer, since a large part of the code that would typically need to be implemented can be covered by our components, and only specific parts need to be implemented by hand. This is possible because the leaflet objects encapsulated by the *Web Components* can be accessed at any time.

The remainder of the paper is organized as follows: In Section II, some related work will be presented. Section III presents the overall architecture before Section IV demonstrates an example application that shows the implemented components in action. Section V concludes the paper with a summary and an outlook on further work in this field.

II. Related work

An approach using the Polymer [6] library for building a web-component interface around the leaflet library was established in 2015 [7]. However, the development seemed stopped in 2016. We follow the approach chosen there in the structural design, but go further in that our components access data provider on the server side. This can be used in particular to implement “live scenarios", as well as to load and display GeoJSON objects from multiple data-sources. Furthermore, the development of leaflet has continued over the last 9 years and our components also use additional packages, such as hierarchical clustering of markers at larger scales to avoid cluttering the map view [8]. The DB-Web Components [9] that we developed last year follow an analogous low-code approach in which relational database content is embedded declaratively in HTML pages.

III. Architecture

The architecture of our application is shown in Figure 1. On the client side, our components run within a browser. There they are responsible for displaying maps, overlays, and geographic features, such as markers, lines, etc. A number of components can optionally obtain their information from data sources on the server side. For example, the marker component can cyclically read its position from a time series database, or our GeoJSON component can access one or more `FeatureSets` and thus visualize many objects of different types with a single instruction.

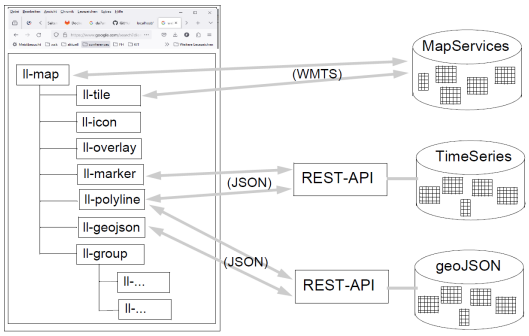

Figure 1. Architecture.

Next, we will show by way of example how the components can be used to implement a sample application.

IV. Example

Figure 2 shows the implemented application. On it, two paddlers can be seen paddling towards each other to enjoy a beer together at the picnic spot (represented by a beer icon). The associated application code can be seen in Figure V. The basemap and an additional alternative ESRI satellite image are defined between line 2 and line 12 in the context of the `ll-map`, `ll-tile`, and `ll-overlay` component. The `ll-overlay` element starting at line 9 defines that an additional overlay, labeled "Sea layer", with nautical symbols is displayed on top of the basemap.

The `ll-geojson` element on line 13 loads the data about the portages (location where the boat has to be transported over land) from a server-side `geojson` data source specified by the `url` parameter. In addition, an icon symbol (a paddler carrying his boat overhead) is specified using the `icon-url` and `icon-size` parameters.

In line 19, the picnic spot is defined at a fixed point specified by longitude and latitude. It also has its own icon and a tooltip text that is displayed when the icon is clicked.

The two paddlers to be displayed are defined in lines 25 and 31. Instead of hard-coding the position of the two paddlers as at the picnic area, you can get their current position cyclically from a time series database, which is specified by the parameter `url`. The two further parameters `lat-path` and `lng-path` specify where the information about longitude and latitude is located in the *json* response.

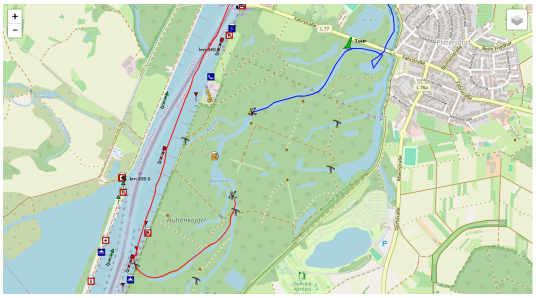

Figure 2. Screenshot of Example Application

Finally, the element `ll-group` is to be explained. As the name suggests, it functions as a grouping element and groups the child elements, which can then be switched on and off using the *layer control* at the top right. This can be seen in Figure 3, where the opened layer control is visible at the top right.

Figure 3. Open *layer control* (top right), which handles the selection of the basemap ("Esri Satelitte"), as well as the overlays ("Sea layer") and groups ("Portages [Img]", "Paddle tour ...") to be shown.

In the figure, the alternative Esri satellite image was selected as the basemap and the *sea layer* with the nautical signs is turned off. The `ll-geojson` data source (portages) also appears in the layer control under the specified label.

V. Conclusion and Outlook

We have implemented a first prototype based on the Lit Framework [10]. In addition to the components presented in the example, there is also an `ll-polyline` component for displaying lines and an `ll-icon` component for the

```
<body>
  <ll-map id="mymap"
          zoom="15">
    <ll-tile url='http://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/\
                  MapServer/tile/{z}/{y}/{x}'
                  label="Esri␣Satellite">
               &copy; <a href="http://www.esri.com/">Esri</a>
    </ll-tile>
    <ll-overlay url="http://t1.openseamap.org/seamark/{z}/{x}/{y}.png"
                label="Sea␣layer">
              &copy; <a href="http://t1.openseamap.org/copyright">OpenSeaMap</a>
    </ll-overlay>
    <ll-geoJSON label="Portages␣<img␣src='icons/portage.png'␣width='15'>"
                icon-url="./icons/portage.png"
                url="http://localhost/llwc/readGeoJSON.php?file=data/portages.json">
                path="result">
    </ll-geoJSON>
    <ll-group label="Paddle␣tour␣February␣22,␣2024">
      <ll-marker lat="48.8776"
                 lng="8.1339"
                 icon-size="20"
                 icon="icons/beer.png"
                 tooltip="Picnic<br>Place">
      </ll-marker>
      <ll-marker icon="icons/kanu2.png" icon-size="20"
                 tooltip="Thomas"
                 url="http://localhost/llwc/readFromInflux.php/thomas?num=1"
                 lat-path="result[0].lat"
                 lng-path="result[0].lon">
      </ll-marker>
      <ll-marker icon="icons/kanu.png" icon-size="20"
                 tooltip="Andreas"
                 url="http://localhost/llwc/readFromInflux.php/andreas?num=1"
                 lat-path="result[0].lat"
                 lng-path="result[0].lon">
      </ll-marker>
    </ll-group>
  </ll-map>
</body>
```

Figure 4. Markup code for example application

definition of icon objects, which can then be referenced by other components.

Further work is planned in the area of simplified integration of JavaScript functions into the leaflet event mechanism, as well as the processing of feature information when representing *GeoJSON* objects.

References

[1] OpenStreetMap, "OpenStreetMap", Last accessed 17.1.2025, 2024, [Online]. Available: https://www.openstreetmap.org.

[2] V. Agafonkin, "Leaflet api reference", Last accessed Last accessed 17.1.2025, 2024, [Online]. Available: https://leafletjs.com/reference.html.

[3] OpenSeaMap, "The free nautical chart", Last accessed 17.1.2025, 2024, [Online]. Available: https://openseamap.org/index.php?id=openseamap&L=1.

[4] H. Butler, S. Gillies, and T. Schaub, "Rfc 7946 - the geojson format", Last accessed 17.1.2025, 2016, [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7946.

[5] webcomponents.org, "WebComponents - Specifications", Last accessed 17.1.2025, 2024, [Online]. Available: https://www.webcomponents.org/specs.

[6] POLYMER, "Polymer library", Last accessed 17.1.2025, [Online]. Available: https://polymer-library.polymer-project.org/.

[7] Hendrik Brummermann et al., "Leaflet-map", Last accessed 17.1.2025, 2015, [Online]. Available: https://github.com/leaflet-extras/leaflet-map.

[8] Erik Nikulski, "Leaflet.markercluster", Last accessed 17.1.2025, 2024, [Online]. Available: https://github.com/Leaflet/Leaflet.markercluster.

[9] A. Schmidt and T. Münch, "Web Components for Database Developers", in *Proceediungs of the Sixteenth International Conference on Advances in Databases, Knowledge, and Data Applications*, (Athen, Griechenland, Mar. 10–14, 2024), 2024, pp. 20–22.

[10] LIT, "Simple. Fast. Web Components", Last accessed 17.1.2025, 2024, [Online]. Available: https://lit.dev/.