# **Libraries used:**

1. MapboxGL JS: Used to run interactive map. Please see Mapbox examples here: <https://docs.mapbox.com/mapbox-gl-js/example/>
2. Jquery
3. Html2canvas: Used to convert HTML elements to Canvas. Canvas can be exported to JPG format.
4. JSPDF: Used to convert JPG canvas to PDF. <http://raw.githack.com/MrRio/jsPDF/master/docs/index.html>
5. Mapbox Geocoder API: <https://github.com/mapbox/mapbox-gl-geocoder> used to search locations and places within customized bounds.
6. Jquery-confirm: Used to display prompt dialog for entering Title for PDF document. <https://craftpip.github.io/jquery-confirm/>

# **CSS used:**

1. W3CSS: <https://www.w3schools.com/w3css> for better design.

# **Layers used:**

Layers used are those from Leallet.gpkg which was imported to QGIS and then reconverted and exported to GEOJSON. Each layer has a Geojson file (11 in total). Geojson does not include style for layers, so we used styling directly in MapboxGL JS like in this script code:

map.addSource('leaflet\_address', {type: 'geojson', data: 'leaflet\_address.geojson'});

// The script code above adds new source to the web map, which is a Geojson source for Addresses.

map.addLayer({'id': 'leaflet\_address', 'type': 'line', 'source': 'leaflet\_address', 'paint': {'line-width': 1.26000,'line-color': '#3c90f1'},'layout': {'visibility': 'visible'}});

// The script code above should be added to add a new layer to the web map, which gets its data from written source (addSource).  
The paint option includes line-width, line-color and visibility. All which can be changed or add new paint or layout options as described here in the documentation: <https://docs.mapbox.com/mapbox-gl-js/style-spec/layers/>

A layer type can be: fill (polygon), line, symbol (icon or text), raster, circle or heatmap. We used symbol-text type for labels as in this script code:

map.addLayer({'id': 'leaflet\_parcel-dims-label', 'type': 'symbol', 'source': 'leaflet\_parcel-dims', 'layout': {'text-field': ['get', 'new\_length'], 'text-variable-anchor': ['top', 'bottom', 'left', 'right'], 'text-radial-offset': 0.5, 'text-justify': 'auto', 'text-size': 12, 'visibility': 'visible'}, 'paint': {'text-color': '#000000'}, 'minzoom': 17});

// Here we see that text-field (meaning text content to be displayed) gets its value from ‘new\_length’ attribute in Geojson file. Each feature in leaflet\_parcel-dims.geojson has its own ‘new\_length’ value. So this layer is only about displaying labels for leaflet\_parcel-dims.geojson features.

# **Canvas to PDF:**

We used a function on clicking on PDF button, that hides first the left panels: $('.changestyle').css('display', 'none'); This script code changes the display CSS attribute to none, means not displayed or hidden.  
$('#footer').css('display', 'block'); The Div footer was hidden at first, but when clicking on PDF to export to PDF, this footer is now displayed using block attribute (default value, meaning not hidden or visible).  
Also, we should hide zoom controls and search box:  
$('.mapboxgl-ctrl-top-right').css('display', 'none');  
$('.mapboxgl-ctrl-bottom-left').css('display', 'none');  
$('.mapboxgl-ctrl-bottom-right').css('display', 'none');

This above script means that all map controls (top right, bottom left and bottom right) are hidden.

Now using Jquery-confirm, we display a prompt dialog asking for Title to be displayed in the footer. After getting the title value, it’s written directly on the div using $("#titlee").html("<b>TITLE: </b>" + title); Please note that the dialog can also be customized following this documentation: <https://craftpip.github.io/jquery-confirm/>

After that, we use html2canvas function to convert the div container for both map and footer, which we named ‘almightycontainer’:  
html2canvas(document.querySelector("#almightycontainer")).then(function (canvas) {all = canvas.toDataURL('image/jpeg'); pdf = new jsPDF({orientation: "landscape", unit: 'pt', format: [canvas.width, canvas.height]}); pdf.addImage(all, 'JPEG', 0, 0, canvas.width, canvas.height); pdf.save("Project.pdf")});

So, the variable ‘all’ is the canvas to image. The variable pdf is calling a new jsPDF with options such as orientation (here landscape), unit (here point, can be px), format (here we used full canvas width and height). pdf.addImage is adding our canvas image to a PDF which will be automatically downloaded as Project.pdf.  
Please refer to jsPDF documentation for more options: <http://raw.githack.com/MrRio/jsPDF/master/docs/index.html> I should say there are lot of options for a more customized PDF.

After that, we should make hidden controls visible again, and hide the footer. So we do the exact opposite of what was done:  
$('.changestyle').css('display', 'block');  
$('.layercheckbox').css('display', 'block');  
$('.mapboxgl-ctrl-top-right').css('display', 'block');  
$('.mapboxgl-ctrl-bottom-left').css('display', 'block');  
$('.mapboxgl-ctrl-bottom-right').css('display', 'block');  
$('#footer').css('display', 'none');

The footer div is at line 53, it can be customized of course:  
<div id="footer" style="width:100%; height:16%; position: absolute; bottom:0; left:0; background-color: rgba(255, 255, 255, 0.8); z-index: 3; display:none">  
<div class="w3-cell-row">  
<div class="w3-container w3-cell w3-border" style="width:350px">  
<p style="font-size:8px"><b>NOTICE:</b> NO WARRANTY OF ACCURACY. THE INFORMATION SHOWN ON THIS MAP WAS COMPILED FOR USE BY THE CITY OF MOUNTLAKE TERRACE, ITS EMPLOYEES AND CONSULTANTS. THE CITY OF MOUNTLAKE TERRACE DOES NOT WARRANT THE ACCURACY OF ANYTHING SET FORTH IN THIS MAP. NEITHER THE CITY OF MOUNTLAKE TERRACE NOR ITS EMPLOYEES OR REPRESENTATIVES SHALL BE LIABLE FOR THE INFORMATION GIVEN ON THIS MAP, NOR FOR ANY ORAL REPRESENTATION PROVIDED BASED ON SAID MAP</p>  
</div>  
<div class="w3-container w3-cell w3-border">  
<p class="w3-center" id="titlee"><b>TITLE: </b></p>  
<div style="font-size:10px"><img src="811.png" style="width:30px"> CALL BEFORE YOU DIG: 1-800-424-5555 (OR 811)</div>  
</div>  
<div class="w3-container w3-cell w3-border w3-center" style="">  
<div><img src="logo.png" style="width:120px"></div>  
<div id="scalecontrol" class="w3-center" style="margin-top: 20px; margin-left: 20px"></div>  
</div>  
 </div>  
</div>

Each ‘w3-cell’ class div inside a ‘w3-row’ class div, is a cell from left to right in a row. The first cell is the notice, the second is the one containing the title and 811 logo, the third one is the one containing the logo and scale control.  
Note that the scale control has an ID ‘scalecontrol’ and is empty for the moment, but it should be populated with Mapbox scale control with this script code at line 694:  
map.addControl(new mapboxgl.ScaleControl({unit: 'imperial'}));

The unit can be imperial or metric or nautical. Full reference here: <https://docs.mapbox.com/mapbox-gl-js/api/markers/#scalecontrol>

Always available if you have any questions in MapboxGL JS (in general) or JSPDF etc.. or anything else 😊  
Also, notice how MapboxGL is very easy to use and have a lot more options (they recently introduced globe projection, which is very cool. You can discover its abilities here: https://docs.mapbox.com/mapbox-gl-js/example/  
Also, you do not need to rely on Google base maps anymore since Mapbox Studio allows to create customized web base maps: <https://www.mapbox.com/mapbox-studio> and import your style to MapboxGl Js directly).  
Thanks.