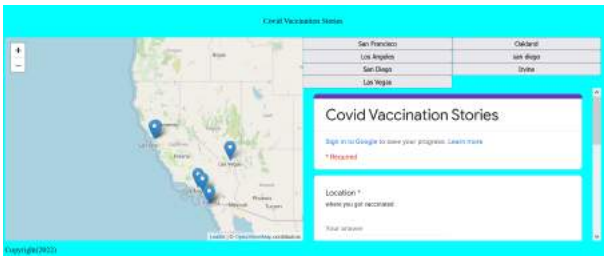
Design DeCSSions

Adding the data from our survey into our mapplication!



Goals

- Add content from Google Form into a `div`
- Use CSS Grid to change page layouts

Today's lab will cover how to put your data in other places besides just the map and work with CSS Grid layouts.

Start by creating a `week6` folder in your lab assignments repo.

Get ahead start

If you finished `lab 5`, you can also copy the contents of your `week_5` folder and skip the following setup section.

Starting template code for lab #6

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Lab 6</title>
        <!-- hint: remember to change your page title! -->
        <meta charset="utf-8" />
```

```
        <link rel="shortcut icon" href="#">
        <link rel="stylesheet" href="styles/style.css">

        <!-- Leaflet's css-->
        <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

        <!-- Leaflet's JavaScript-->
        <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js">
</script>

        <!-- Papa Parse -->
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js">
</script>
    </head>

    <body>
        <header>
            <!-- space for a menu -->
        </header>

        <div class="main">
            <div id="contents">
                <!-- Be sure to use your own survey here!!!!!!! -->
                <iframe
src="https://docs.google.com/forms/d/e/1FAIpQLSfcElv5dlXInR7XHQz27_OcYJlWcIUr-
GBbc-ocefWlGd1uXg/viewform?embedded=true" width="640" height="475"
frameborder="0" marginheight="0" marginwidth="0">Loading…</iframe>

            </div>
            <div id="the_map"></div>
        </div>
        <div id="footer">
            Copyright(2023)
        </div>
        <script src="js/init.js"></script>
    </body>
</html>
```

js/init.js

```
// declare variables
let mapOptions = {'center': [34.0709,-118.444],'zoom':5}

// use the variables
const map = L.map('the_map').setView(mapOptions.center, mapOptions.zoom);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
```

```
}).addTo(map);

// create a function to add markers
function addMarker(lat,lng,title,message){
    console.log(message)
    L.marker([lat,lng]).addTo(map).bindPopup(`<h2>${title}</h2>
<h3>${message}</h3>`)
    return message
}

const dataUrl = "https://docs.google.com/spreadsheets/d/e/2PACX-
1vSNq8_prhrSwK3CnY2pPptqMyGvc23Ckc5MCuGMMKljW-dDy6yq6j7XAT4m6GG69CISbD6kfBF0-
ypS/pub?output=csv"

function loadData(url){
    Papa.parse(url, {
        header: true,
        download: true,
        complete: results => processData(results)
    })
}

function processData(results){
    console.log(results)
    results.data.forEach(data => {
        console.log(data)
        addMarker(data.lat,data.lng,data['Where did you get
vaccinated?'],data['Have you been vaccinated?'])
    })
}

loadData(dataUrl)
```

styles/style.css

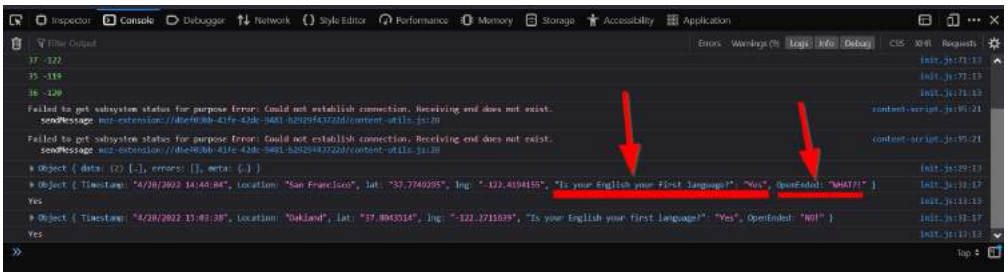
```
body{
    display: grid;
    grid-auto-rows: auto 1fr;
    grid-template-areas: "header" "main_content" "footer";
    background-color: aqua;
}

header{
    grid-area: header;
}

#footer{
    grid-area: footer;
}

.main{
    grid-area: main_content;
```

```
    grid-template-columns: 1fr 1fr;
    grid-template-areas: "main_map content";
    display: grid;
}

#contents{
    grid-area: content;
}

#the_map{
    height:80vh;
    grid-area: main_map;
}
```

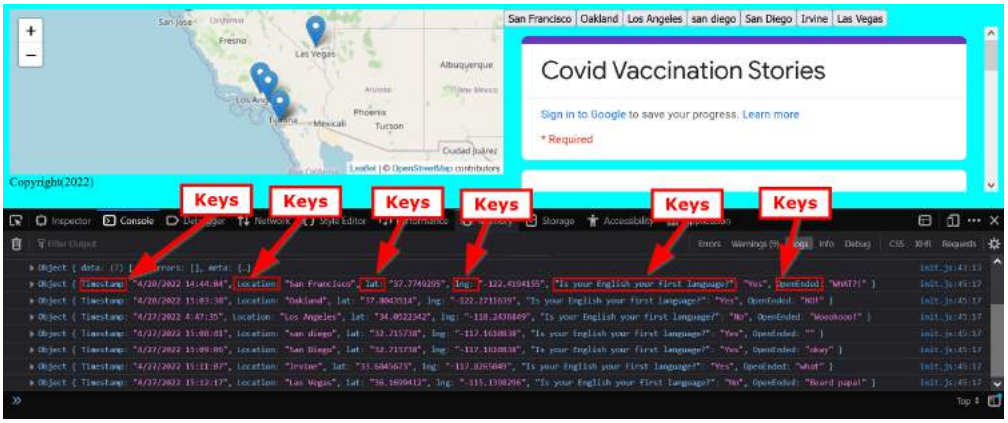
Last update: 2023-05-11

Getting our data in place!

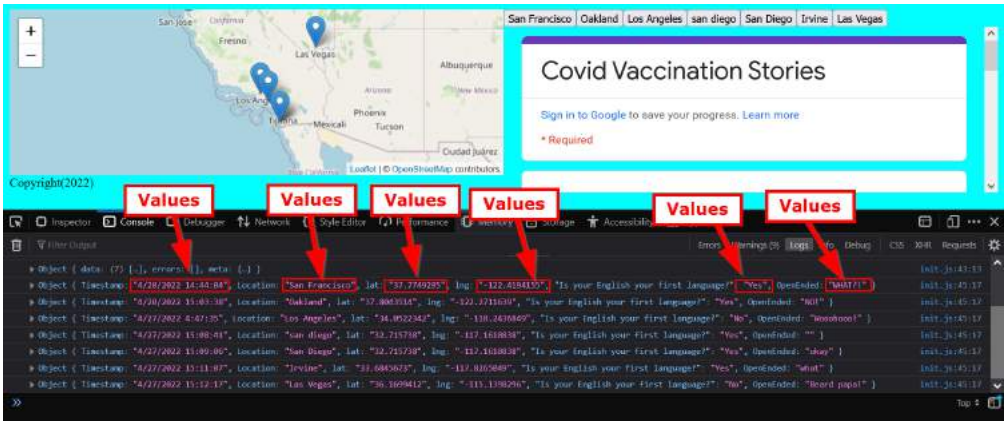
Go ahead and start the live server to make sure that the code is up and running. In Firefox, open up the `Debug Console` (`Right Button` then click inspect element OR press `F12`) and click on the array from last week:



Remember that in an JSON object, `keys` are essentially the **field names**:



And `values` are the **contents**:



Remember to access a value, you have to access the `object` and then the `key` using `.` **dot notation** or the `['name of field']` **bracket notation**.

For example, last lab when we wanted the `lat` data, we used `data.lat`:

```
addMarker(data.lat,data.lng,data['Where did you get vaccinated?'],data['Have you been vaccinated?'])
```

Prepping the `div`s

🏃 Warm-up: Add a `div` for the survey

Before we do anything to new, let's move our survey into its own `div`!

Answer

index.html

```
    <div id="contents">
        <div id="theSurvey">
            <iframe
src="https://docs.google.com/forms/d/e/1FAIpQLScD0IOr_U4r0q4HlBkZ7olkA5OJpgInePF8DQbI
embedded=true" width="640" height="720" frameborder="0" marginheight="0"
marginwidth="0">Loading…</iframe>
        </div>
    </div>
```

Let's also change the `height` and `width` to `100%` so the `iframe` can finally follow our CSS rules!

```
<iframe
src="https://docs.google.com/forms/d/e/1FAIpQLScD0IOr_U4r0q4HlBkZ7olkA5OJpgInePF8DQb
embedded=true" width="100%" height="100%" frameborder="0" marginheight="0"
marginwidth="0">Loading…</iframe>
```

With the survey in its own `div` now, we can then add a `<div>` for our new content:

index.html

```
        <div id="contents">
            <div id="placeForButtons">
            <div id="theSurvey">
                <iframe
```

```
src="https://docs.google.com/forms/d/e/1FAIpQLScD0IOr_U4r0q4HlBkZ7olkA5OJpgInePF8DQb
embedded=true" width="100%" height="100%" frameborder="0" marginheight="0"
marginwidth="0">Loading…</iframe>
            </div>
        </div>
```

Adding a `function` again?

Now, let's look at the `location` key. We probably want to add the locations to our contents to see what locations or showing up, as some people might put the same location more than twice!

We probably want to make buttons from these locations soooo....

Warm-up question: Create buttons (call-back to lab #2)!!

Question

Which line should we put this create buttons function: `createButtons(lat,lng,location);`

Answers

It should go in our `addMarker()` function! So that when we add markers, the `buttons` are added afterwards!

Add the line for the `createButton()` function in our JavaScript file:

js/init.js

```
function addMarker(lat,lng,title,message){
    // console.log(message)
    L.marker([lat,lng]).addTo(map).bindPopup(`<h2>${title}</h2> <h3>${message}
</h3>`)
    createButtons(lat,lng,location)
    return message
}
```

Refactor the `addMarker()`

Notice how the `addMarker()` has a lot of parameters now? Like, 4? That's quite a lot, and will be harder to manage if we want to add more.

So, let's pass in the whole `Object` instead of pieces of the object like so: `addMarker(data)`

Before proceeding to the exercise make sure your `addMarker()` call looks like the following inside the `processData()` function:

js/init.js

```
function processData(results){
    console.log(results)
    results.data.forEach(data => {
        console.log(data)
        addMarker(data)
    })
}
```

⚽ In-class Exercise #1 - Refactor the `addMarker()` function

We are refactoring the `addMarker()` function during our loop to take in the whole object as a parameter. How might we re-write our `addMaker()` function to take in the object?

Tasks

1. Re-write the `addMarker()` function to take in the data object from the `forEach()` loop.

Answer

js/init.js

```
function addMarker(data){
    // console.log(data['Where did you get vaccinated?'])
    L.marker([data.lat,data.lng]).addTo(map).bindPopup(`<h2>${data['Where did you
get vaccinated?']}</h2> <h3>${data['Have you been vaccinated?']}</h3>`)
    createButtons(data.lat,data.lng,data['Where did you get vaccinated?'])
    return data['Where did you get vaccinated?']
}
```

Important note!

1. Make sure you changed the `return message` line in the end of the function, otherwise it will break since it `message` is not defined anywhere!
2. Make sure we use `data` object! Which means making sure we use the `object` + `.` + `key` format.

Reminder about accessing `object-keys` again

Remember, since our data is being stored in an object, the `createButton()` function should look like this:

```
createButtons(data.lat,data.lng,data.location)
```

> - Where we are accessing the `data` `object`'s `lat`, `lng`, and `location`.

Note

Your `keys==` **MUST** always match your data == `object`, you can use the console to check!!

For example: If your survey spreadsheet has `latitude` instead of `lat` then the you MUST use `data.latitude`

And now, just like in lab 3, we are going to add buttons!

Here's the `createButton()` from lab 3:

```
function createButtons(lat,lng,title){
    const newButton = document.createElement("button"); // adds a new button
    newButton.id = "button"+title; // gives the button a unique id
    newButton.innerHTML = title; // gives the button a title
    newButton.setAttribute("lat",lat); // sets the latitude
    newButton.setAttribute("lng",lng); // sets the longitude
    newButton.addEventListener('click', function(){
        map.flyTo([lat,lng]); //this is the flyTo from Leaflet
    })
    document.body.appendChild(newButton); //this adds the button to our page.
}
```

Wait.. it didn't work? Well, that's because we have to tweak a few things…

1. Next we need to change the `document.body.appendChild(newButton)` to use the `div` that we created earlier!

Adding buttons to our `div`

To address the second issue of targeting our `div`, we need to utilize the JavaScript method of selecting Elements called:

`getElementById()`

Learn more about `getElementById()`

Just running the method doesn't do anything, so we need to store it in a variable:

```
const spaceForButtons = document.getElementById('placeForButtons')
```

Remember the `appendChild()` that adds content? We will use that method to add our button to our `spaceForButtons` variable that specifies the `div`:

```
spaceForButtons.appendChild(newButton);
```

The final `createButtons()` and `addMarker()` functions should look like this:

```
function addMarker(data){
        // console.log(data)
        // these are the names of our lat/long fields in the google sheets:


        L.marker([data.lat,data.lng]).addTo(map).bindPopup(`<h2>${data['Where did
you get vaccinated?']}</h2> <h3>${data['Have you been vaccinated?']}</h3>`)
        // adding our create button function
        createButtons(data.lat,data.lng,data.location)
        return data.Location
}

function createButtons(lat,lng,title){
    const newButton = document.createElement("button"); // adds a new button
    newButton.id = "button"+title; // gives the button a unique id
    newButton.innerHTML = title; // gives the button a title
    newButton.setAttribute("lat",lat); // sets the latitude
    newButton.setAttribute("lng",lng); // sets the longitude
    newButton.addEventListener('click', function(){
        map.flyTo([lat,lng]); //this is the flyTo from Leaflet
    })
    const spaceForButtons = document.getElementById('placeForButtons')
```

```
    spaceForButtons.appendChild(newButton);//this adds the button to our page.
}
```

Horrah!!

Our current HTML structure

Here's a quick schematic of our HTML and CSS:

```
HTML
  └BODY
     └.main
        ├#contents
        │    ├#placeForButtons
        │    └#survey
        └#the_map
```

Check to see if your JavaScript and HTML code is similar to the following before moving on.

🏁 Check point

js/init.js

```
// declare variables
let mapOptions = {'center': [34.0709,-118.444],'zoom':5}

// use the variables
const map = L.map('the_map').setView(mapOptions.center, mapOptions.zoom);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);

function addMarker(data){
    // console.log(data)
    // these are the names of our lat/long fields in the google sheets:
    L.marker([data.lat,data.lng]).addTo(map).bindPopup(`<h2>${data['Where did
you get vaccinated?']}</h2> <h3>${data['Have you been vaccinated?']}</h3>`)
    createButtons(data.lat,data.lng,data['Where did you get vaccinated?'])
    return
}

function createButtons(lat,lng,title){
    const newButton = document.createElement("button"); // adds a new button
    newButton.id = "button"+title; // gives the button a unique id
    newButton.innerHTML = title; // gives the button a title
```

```
    newButton.setAttribute("lat",lat); // sets the latitude
    newButton.setAttribute("lng",lng); // sets the longitude
    newButton.addEventListener('click', function(){
        map.flyTo([lat,lng]); //this is the flyTo from Leaflet
    })
    const spaceForButtons = document.getElementById('placeForButtons')
    spaceForButtons.appendChild(newButton);//this adds the button to our
page.
}

const dataUrl = "https://docs.google.com/spreadsheets/d/e/2PACX-
1vSNq8_prhrSwK3CnY2pPptqMyGvc23Ckc5MCuGMMKljW-dDy6yq6j7XAT4m6GG69CISbD6kfBF0-
ypS/pub?output=csv"

function loadData(url){
    Papa.parse(url, {
        header: true,
        download: true,
        complete: results => processData(results)
    })
}

function processData(results){
    console.log(results)
    results.data.forEach(data => {
        console.log(data)
        addMarker(data)
    })
}

loadData(dataUrl)
```

index.html

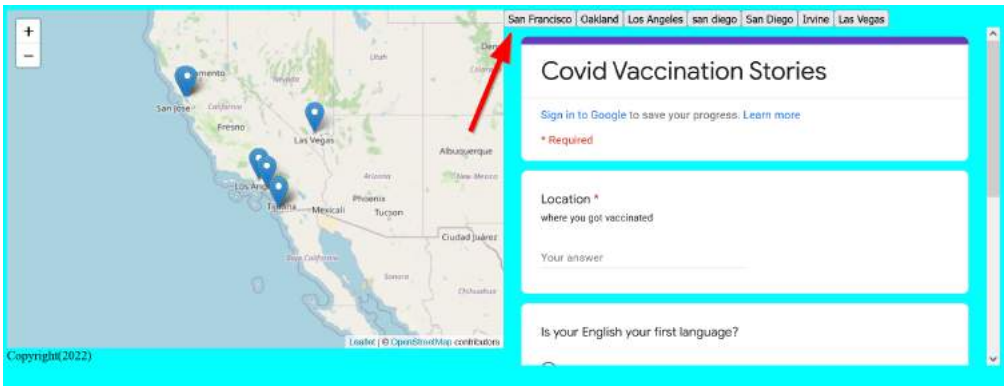
```
<!DOCTYPE html>
<html>
    <head>
        <title>Lab 6</title>
        <!-- hint: remember to change your page title! -->
        <meta charset="utf-8" />
        <link rel="shortcut icon" href="#">
        <link rel="stylesheet" href="styles/style.css">

        <!-- Leaflet's css-->
        <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

        <!-- Leaflet's JavaScript-->
        <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js">
</script>

        <!-- Papa Parse -->
```

```
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js">
</script>
    </head>

    <body>
        <header>
            <!-- space for a menu -->
        </header>

        <div class="main">
            <div id="contents">
                <div id="placeForButtons">
                <div id="theSurvey">
                    <iframe
src="https://docs.google.com/forms/d/e/1FAIpQLScD0IOr_U4r0q4HlBkZ7olkA5OJpgIneP
embedded=true" width="100%" height="100%" frameborder="0" marginheight="0"
marginwidth="0">Loading…</iframe>
                </div>
            </div>
            <div id="the_map"></div>
        </div>
        <div id="footer">
            Copyright(2022)
        </div>
        <script src="js/init.js"></script>
    </body>
</html>
```

Last update: 2023-05-11

CSS Grid time!

Ok, now that we got the buttons working, notice how they appear at the top?



This is because we don't have a place for them in our CSS Grid yet!

So time to style our page and make it more presentable!

Open up our `style.css` and find the `main` **class** selector, `.main`:

styles/style.css

```
.main{
    grid-area: main_content;
    grid-template-columns: 1fr 1fr;
    grid-template-areas: "main_map content";
    display: grid;
}
```

Because our `div` sits within the `content` area we should find the the `grid-area` named `content`.

Notice that we `#contents` only has one property, which is the `grid-area: content;`

styles/style.css

```
#contents{
    grid-area: content;
}
```

Who defines `grid-area` names?

We do! `grid-areas` are just names for areas we decided for ourselves. If you don't use `grid-areas`, you can also reference grids by row and column, like `grid-area: 1/2` would be grid row #1 and column #2. If 4 grid line values are provided, then `grid-area` treats them as the following: 1. the first value is `grid-row-start`, what row should this `grid-area` start in 2. the second value is `grid-column-start`, what column should this `grid-area` start in 3. the third value is `grid-row-end`, when should this row end. 4. the fourth value is `grid-column-end`, when should this column end.

Ex: grid-area: 2 / 1 / 2 / 1

Now that we found the parent box, `contents`, we need to enable the power of CSS grid in this `div`, so we have to add the `display: grid;` css property:

styles/style.css

```
#contents{
    grid-area: content;
    display: grid;
}
```

Take note of the `theSurvey` selector, which is within the `.main` class but not affected by the `contents` div, since it sits above it:

```
#theSurvey{
    height:80vh;
    grid-area: main_map;
}
```

Recall the schematic of our HTML and CSS:

```
HTML
  └BODY
     └.main
        ├#contents
        │    ├#placeForButtons
        │    └#theSurvey
        └#the_map
```

Wait a second… There's something missing in our CSS file according to that schematic… Do you know what it is?

Our poor `placeForButtons` and `theSurvey` doesn't have a selector!

Let's fix it by adding a CSS selector for them and a `grid-area: buttonHome`:

```
#theSurvey{
    grid-area: survey; (1)
}

#placeForButtons{
    grid-area: buttonHome; (2)
}
```

1. `grid-area` gives the name to our area `survey`
2. `grid-area` gives the name to our area `buttonHome`

`grid-area` gives the `#placeForButtons` selector the name `buttonHome` that we will use in the `grid-template-areas`.

Assgining Grid-Template Areas

Go back to the `contents` selector add these two css properties: 1. `grid-template-rows` 2. `grid-template-areas` Set grid `template-rows` to 1fr 3fr if you want a bigger space for buttons and set the `grid-template-areas`'s value to `"survey" "buttonHome"`, which is the names we assigned above and the space between them `" "` makes the `survey` and `buttonHome` on seperate rows:

Space between `grid-template-areas`

Make sure you add a space and `"` between the two `grid-template-areas`!! If you fail to do so, then it will be treated as 1 row!!

To recap with `grid-template-areas`: `"item-one item-two"` - one row, two columns. `"item-one" "item-two"` - two rows, one column. `"item-one item-one" "item-two item-three"` - two rows, two columns, and item-one spans two rows!

```
#contents{
    display: grid;
    grid-template-rows: 1fr 3fr;
    grid-template-areas: "survey" "buttonHome"
}
```

Try flipping `survey` and `buttonHome` in the `property value`, what happens?

Answer

The survey showed up on the right-side! Being able to change layouts on-the-fly demonstrates the power of CSS Grid!

Because it makes sense for the buttons to go ontop, let's keep the layout as `grid-template-areas: "buttonHome survey"`:

```
#contents{
    grid-area: content; 1
    display: grid;2
    grid-template-rows: 1fr 3fr;3
    grid-template-areas: "buttonHome" "survey"4
}
```

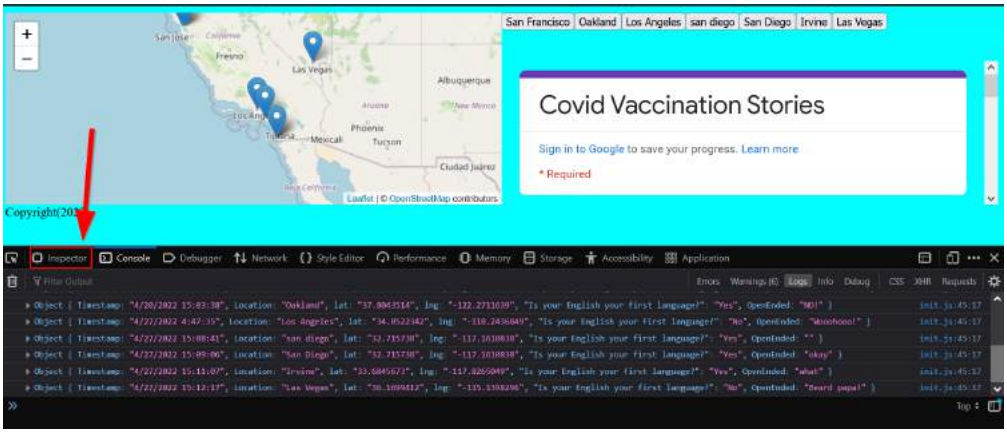
1. `grid-area` gives the name to our area `content` which is used in the `.main` class's `grid.
2. This enables `css-grid` in this `contents` `div`.
3. This sets two rows, one being 25% ontop and 75% below. You can customize this as you'd like with `fr` or `%` or `px` units.
4. This allows us to tell `css-grid` to put the content in the right place.

CSS Grid Debugging in Firefox

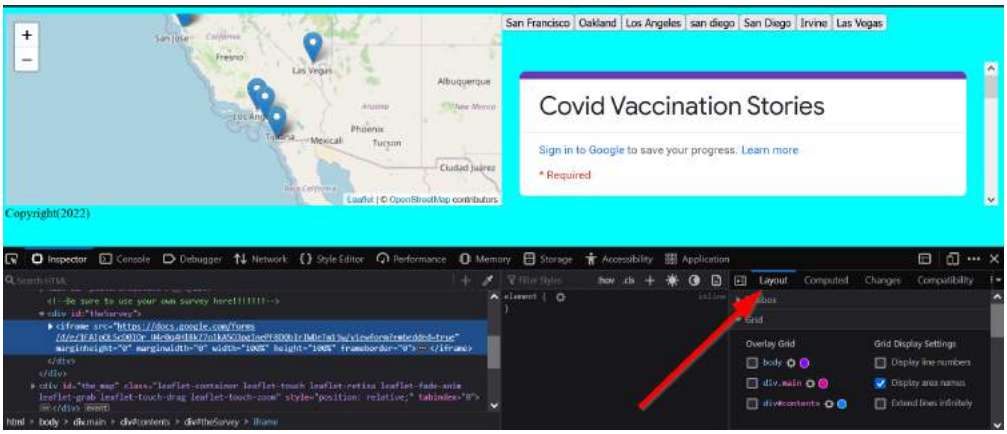
Untested in Chrome or Safari

`CSS grid` is relatively new in Chrome or Safari, so I am not sure how robust the debugging is. I highly recommend using Firefox to troubleshoot `CSS grid`.

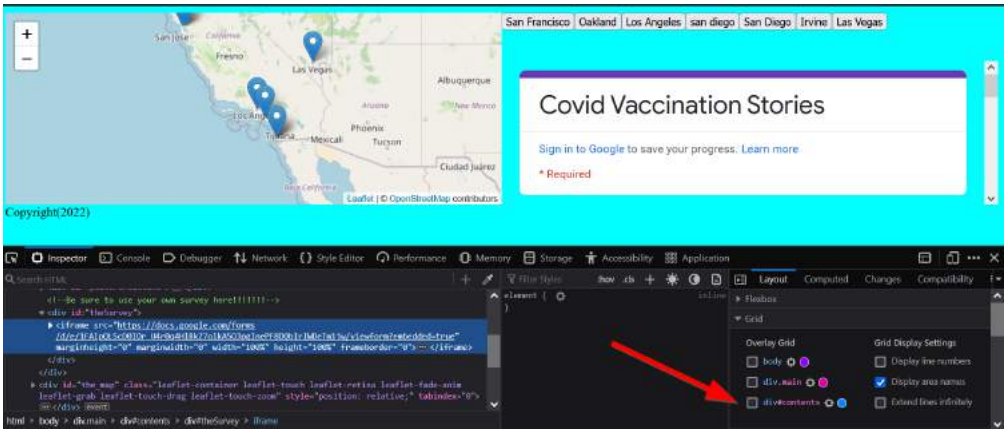
In Firefox inside of your webpage, `Right Button` right click then click on `Inspect Element` and make sure you are on the `Inspector`:



Click on `Layout`:

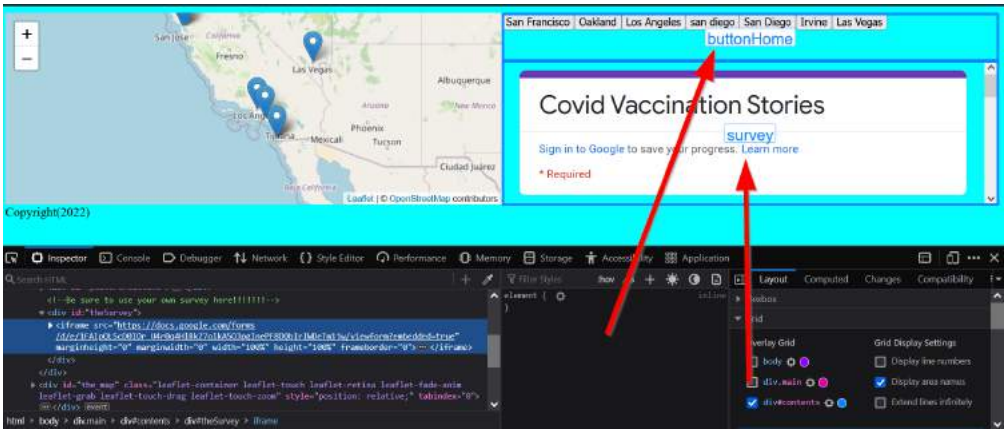


Look for `div#contents`

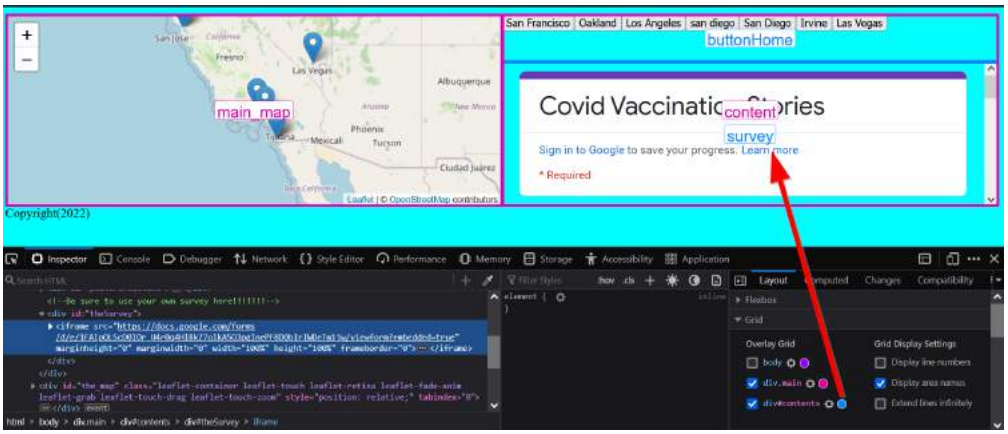


Then make sure `div#contents` and `display area names` are checked:

You will notice that the names of the `grid areas` will display! Very very helpful when debugging CSS Grid!!!

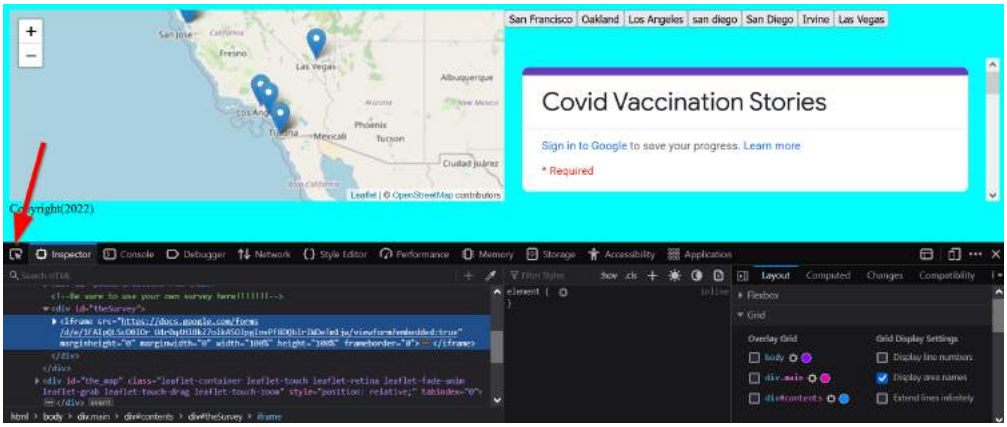


If you check the other `grid` containers like, `div.main`, you can see the multiple grids overlaying each other:

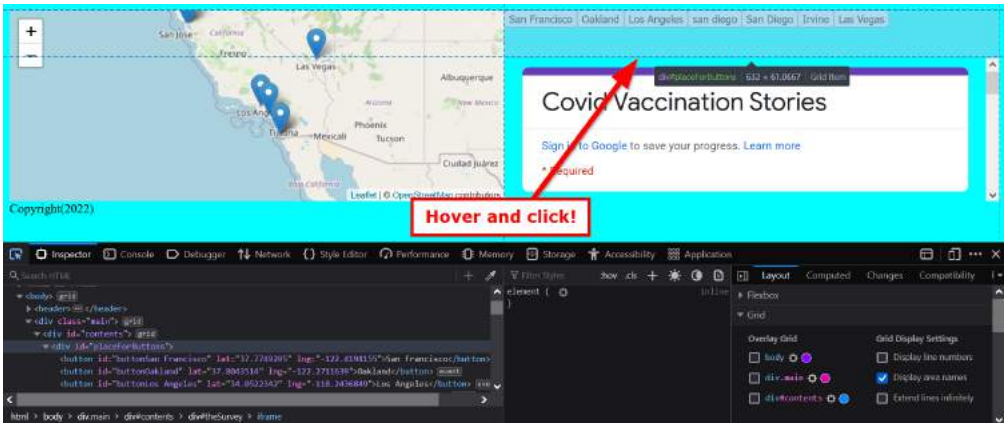


Playing around with the inspector

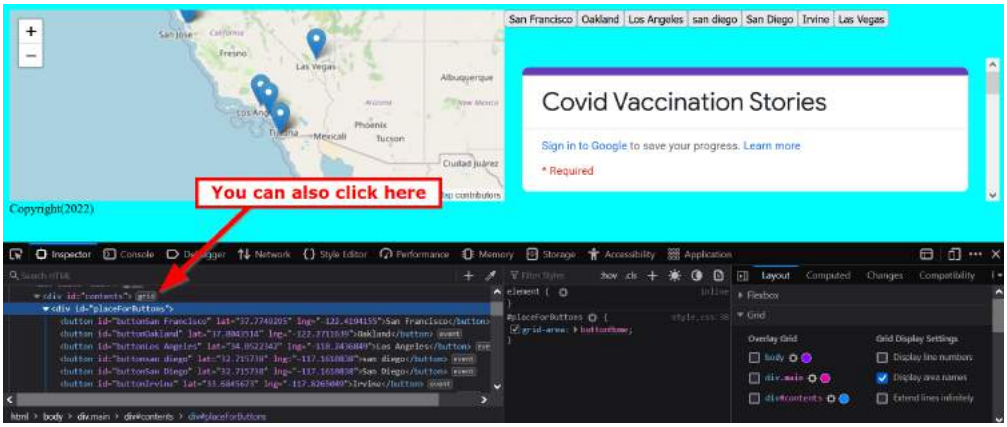
If you click on an element with the `inspector` you can directly edit the CSS:



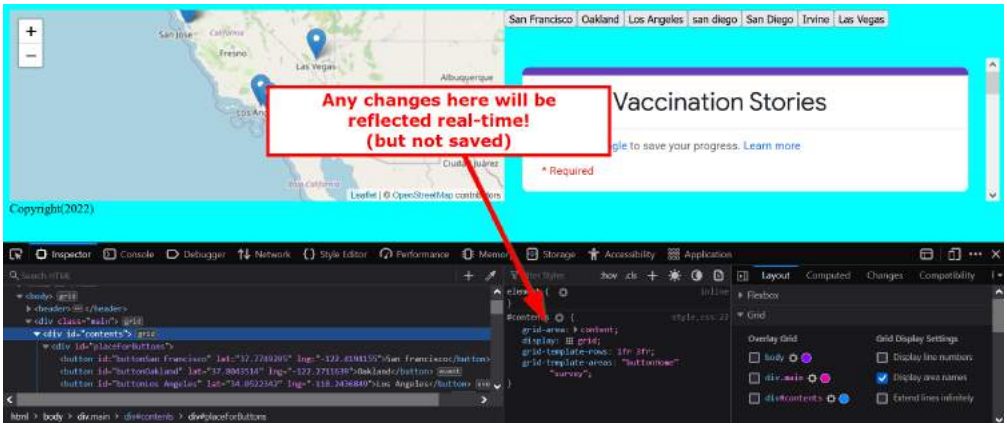
For example, clicking on the `placeForButtons` area allows you to see the CSS styles:



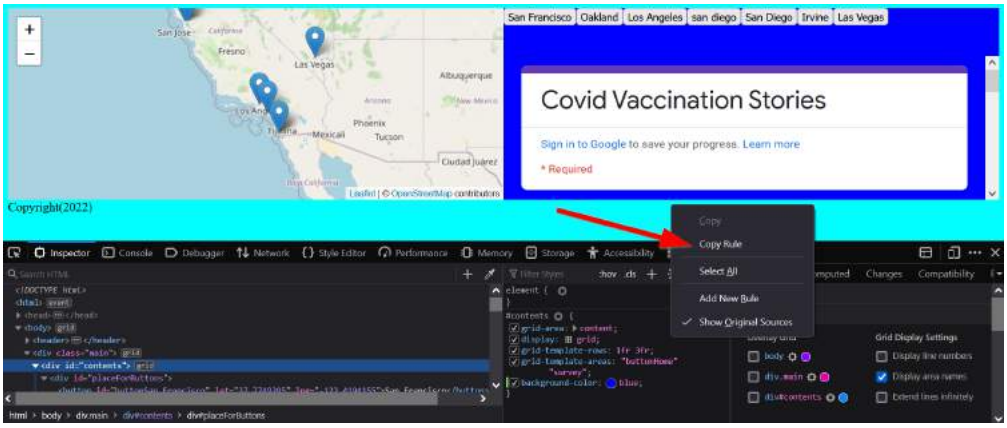
You can also use this to navigate up the HTML tree or find specific elements, like the `contents`:



You can change the styles by editing this box:



If you want to save these changes, you'll have to `right click` and `copy the rule`:



Check point

Check to see if your CSS looks correct before moving on:

styles/style.css

```css
body{
    display: grid;
    grid-auto-rows: auto 1fr;
    grid-template-areas: "header" "main_content" "footer";
    background-color: aqua;
}

header{
    grid-area: header;
}

#footer{
    grid-area: footer;
}

```

```
.main{
    grid-area: main_content;
    grid-template-columns: 1fr 1fr;
    grid-template-areas: "main_map content";
    display: grid;
}

#contents{
    grid-area: content;
    display: grid;
    grid-template-rows: 1fr 3fr;
    grid-template-areas: "buttonHome" "survey"
}

#the_map{
    height:80vh;
    grid-area: main_map;
}

#theSurvey{
    grid-area: survey; /* (1)! */
}

#placeForButtons{
    grid-area: buttonHome; /* (2)! */
}
```

Last update: 2023-05-11

Home C-S-Stretch: Layouts!

Let's add some text to our header in the HTML:

index.html

```
19      <body>
20          <header>
21              COVID Vaccination Stories
22          </header>
```

In the `style.css` file and go to the `body` selector. Then change `grid-template-rows` to the following: `grid-template-rows: 50px auto auto;`.

styles/style.css

```
body{
    display: grid;
    grid-template-rows: 50px auto auto;
    grid-template-areas: "header" "main_content" "footer";
    background-color: aqua;
}
```

This creates a small header ontop that is `50 pixels` tall and `auto` fits the content for the main page and footer.

Adding `gaps` between items

In `CSS-grid` you can add a `gutter` between items that gives a padding in between, as follows:

styles/style.css

```
body{
    display: grid;
    grid-template-rows: 50px auto auto;
    grid-template-areas: "header" "main_content" "footer";
    background-color: aqua;
    gap: 10px;
}
```

> **Vertical scroll bar?**
>
> If you add a gap to the `body` there will be a scroll bar now since it extends the page beyond the initial view port. It is recommended to use gap not with containers that are meant to contain the whole page, like `HTML` or `body`

The `gap` CSS property is very useful for spacing other elements, like buttons!!

Centering content

You can center the header by going to the `header` selector adding the following CSS property `justify-self: center`:

```
header{
    grid-area: header;
    justify-self: center;
}
```

If you want to vertical align, then you need to use `align-self:center`

```
header{
    grid-area: header;
    justify-self: center;
    align-self:center;
}
```

In our `grid-template-columns` we need one more `1fr` for our survey in:

`grid-template-columns: 1fr 1fr 1fr`

Finally, we need to add spaces after the `"header"` and `"footer"` template areas too which indicates which row the template is on:

`grid-template-areas: "header" "mappanel contentpanel" "footer"`

🏁 CSS Check point

Check to see if your `body` and `header` selector looks like the following:

```
body{
    display: grid;
    grid-template-rows: 50px auto auto;
    grid-template-areas: "header" "main_content" "footer";
```

```
    background-color: aqua;
    gap: 10px;
}

header{
    grid-area: header;
    justify-self: center;
    align-self: center;
}
```

Grid-ception and automatically fitting content!

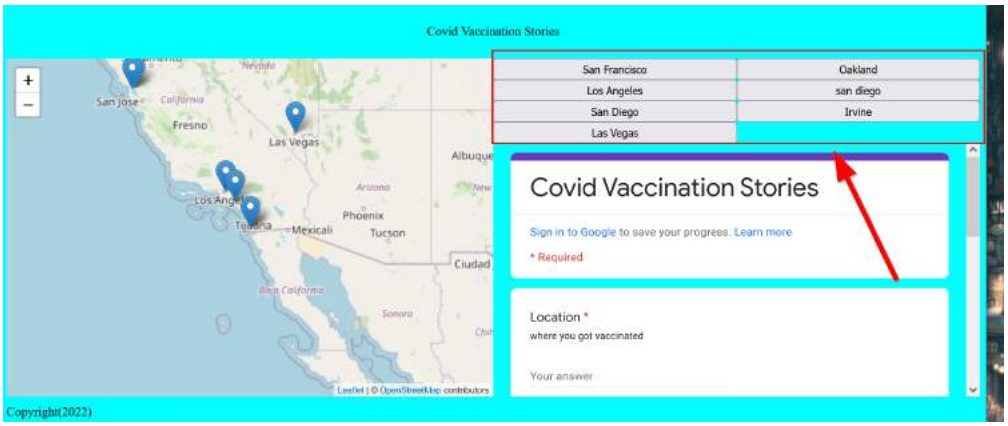
While named `grid-template-areas` can be useful, if you have a lot of content, or if the content might grow in number, you can just specify the number that the content should occupy using the `grid-column` property and `repeat`. Additionally, we can start a `subgrid` within a `grid` element to specify which part of an already existing `grid` we want to visualize.

- More about `repeat`
- More about `subgrid`

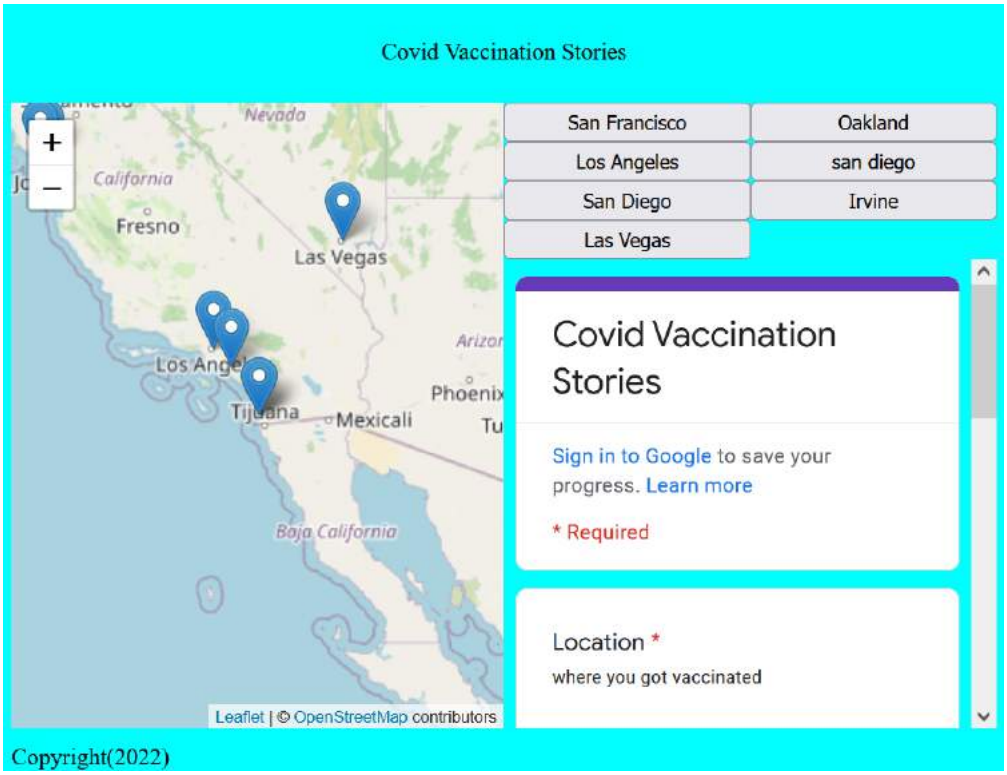
Let's apply a `subgrid` and `repeated columns` of 2 to the `#placeForButtons` selector:

```
#placeForButtons{
    grid-area: buttonHome;
    display:grid;
    grid-template-columns: repeat(2, 1fr);
}
```

This creates a new column after every 2 evenly spaced item!!!

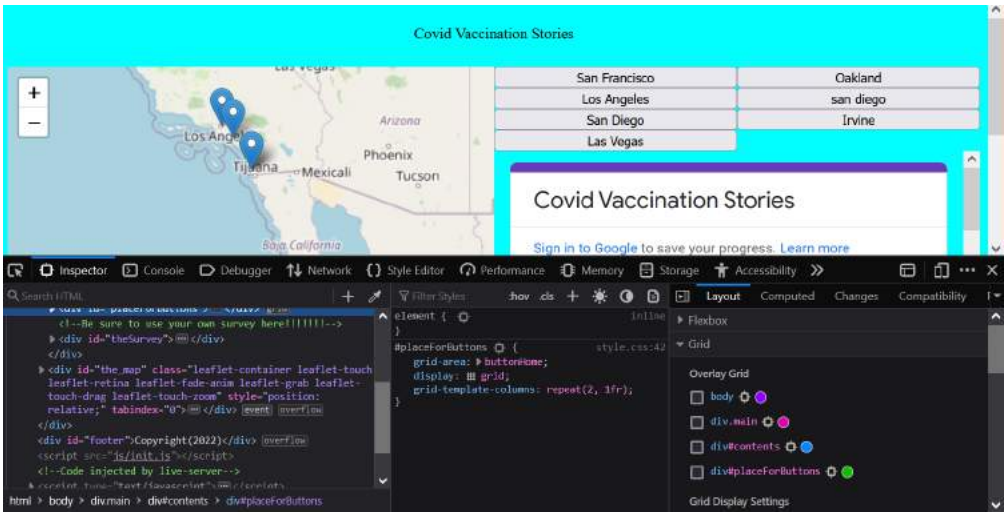


It even works when you resize the page!



So cool!!!

Remember, if you turn on the debugger in Firefox, you can see the multiple grids:



And there you have it! We've only scratched the surface of CSS Grid, but this is enough for you to complete the lab assignment!

Last update: 2023-05-11

✅ Final Template Code

index.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Lab 6</title>
        <!-- hint: remember to change your page title! -->
        <meta charset="utf-8" />
        <link rel="shortcut icon" href="#">
        <link rel="stylesheet" href="styles/style.css">

        <!-- Leaflet's css-->
        <link rel="stylesheet"
href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css" />

        <!-- Leaflet's JavaScript-->
        <script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js">
</script>

        <!-- Papa Parse -->
        <script
src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.0/papaparse.min.js">
</script>
    </head>

    <body>
        <header>
            Covid Vaccination Stories
        </header>

        <div class="main">
            <div id="contents">
                <div id="placeForButtons"></div>
                <!-- Be sure to use your own survey here!!!!!!! -->
                <div id="theSurvey">
                    <iframe
src="https://docs.google.com/forms/d/e/1FAIpQLSfcElv5dlXInR7XHQz27_OcYJlWcIUr-
GBbc-ocefWlGd1uXg/viewform?embedded=true" width="100%" height="100%"
frameborder="0" marginheight="0" marginwidth="0">Loading…</iframe>
                </div>

            </div>
            <div id="the_map"></div>
        </div>
        <div id="footer">
```

```
                Copyright(2023)
            </div>
            <script src="js/init.js"></script>
        </body>
    </html>
```

js/init.js

```
// declare variables
let mapOptions = {'center': [34.0709,-118.444],'zoom':5}

// use the variables
const map = L.map('the_map').setView(mapOptions.center, mapOptions.zoom);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
    attribution: '&copy; <a
href="https://www.openstreetmap.org/copyright">OpenStreetMap</a>
contributors'
}).addTo(map);

function addMarker(data){
    // console.log(data)
    // these are the names of our lat/long fields in the google sheets:
    L.marker([data.lat,data.lng]).addTo(map).bindPopup(`<h2>${data['Where did
you get vaccinated?']}</h2> <h3>${data['Have you been vaccinated?']}</h3>`)
    createButtons(data.lat,data.lng,data['Where did you get vaccinated?'])
    return
}

function createButtons(lat,lng,title){
    const newButton = document.createElement("button"); // adds a new button
    newButton.id = "button"+title; // gives the button a unique id
    newButton.innerHTML = title; // gives the button a title
    newButton.setAttribute("lat",lat); // sets the latitude
    newButton.setAttribute("lng",lng); // sets the longitude
    newButton.addEventListener('click', function(){
        map.flyTo([lat,lng]); //this is the flyTo from Leaflet
    })
    const spaceForButtons = document.getElementById('placeForButtons')
    spaceForButtons.appendChild(newButton);//this adds the button to our
page.
}

const dataUrl = "https://docs.google.com/spreadsheets/d/e/2PACX-
1vSNq8_prhrSwK3CnY2pPptqMyGvc23Ckc5MCuGMMKljW-dDy6yq6j7XAT4m6GG69CISbD6kfBF0-
ypS/pub?output=csv"

function loadData(url){
    Papa.parse(url, {
        header: true,
        download: true,
        complete: results => processData(results)
```

```
    })
}

function processData(results){
    console.log(results)
    results.data.forEach(data => {
        console.log(data)
        addMarker(data)
    })
}

loadData(dataUrl)
```

styles/style.css

```
body{
    display: grid;
    grid-template-rows: 50px auto auto;
    grid-template-areas: "header" "main_content" "footer";
    background-color: aqua;
    gap: 10px;
}

header{
    grid-area: header;
    justify-self: center;
    align-self: center;
}

#footer{
    grid-area: footer;
}

.main{
    grid-area: main_content;
    grid-template-columns: 1fr 1fr;
    grid-template-areas: "main_map content";
    display: grid;
}

#contents{
    grid-area: content;
    display: grid;
    grid-template-rows: 1fr 3fr;
    grid-template-areas: "buttonHome" "survey"
}

#the_map{
    height:80vh;
    grid-area: main_map;
}

```

```
#theSurvey{
    grid-area: survey;
}

#placeForButtons{
    grid-area: buttonHome;
    display:grid;
    grid-template-columns: repeat(2, 1fr);
}
```

Last update: 2023-05-11