LONDON PARKIVE

Build Log

This is a logbook documenting the building process of the London Parkive. The build log was started part way through the project, after I finished experimenting with Leaflet, and MapLibre, two javaScript libraries for developing geographical information systems, and around the time I started developing the actual website. It covers most of the development and part of the design process in detail.

🛠️06.02.24

The HTML file I am working on included the script file and styling so I split this across three files (HTML, CSS and JavaScript). Started working on colouring the parks depending on different properties (see 07.02.24 section for images on that).

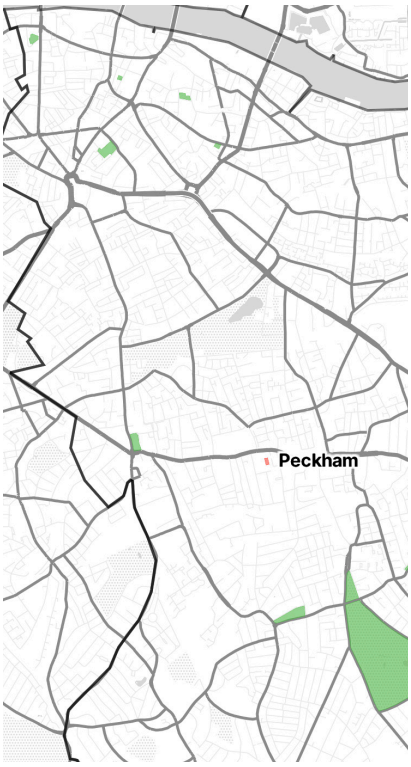
For the altered parks section I was manually writing down the coordinates of the old space using National Library of Scotland's map. I have found this to be very time consuming and easy to make errors (see the image below). So I have decided it might be worth while creating a tool in which I will import the National Library of Scotland's base maps and create a click function that will record the coordinates of the points clicked. This should make the task much more accurate and less time consuming.

This screenshot shows an error that occurred when manually inputting the parks coordinates

🛠️07.02.24

I have finished making three colour schemes for the park polygons on the map. One is for the parks status (whether it is opened or closed), one is for the year opened (the colours of the park are based on which 25 year period they were opened in from 1850s till now) and one is for any alterations that have been made (If a park has remained unchanged, expanded or shrunk.) Below you can see screen shots of each of the colour schemes and the code accompanying them. I was also able to create buttons for each view mode. When a button is clicked the opacity of the layer the button is associated with is set to 0.8 and the other layers opacities are set to 0. (see the code on the following page. Video of this can be viewed on video reel.)

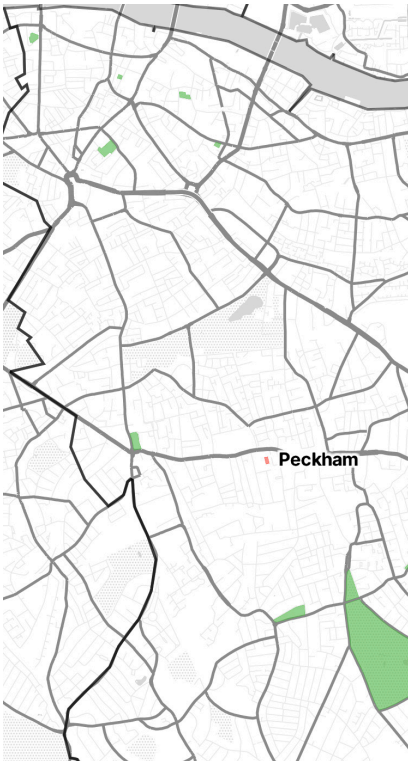
Park Status




```
map.addLayer({
            id : "park_shapes_layer",
            type : "fill",
            source : "parkShapesSource",
            paint : {
                'fill-color' : [
                    'match',
                    ['get', 'status'],
                    'Open', '#50b848',
                    'Defunct', '#ff4c3f',
                    'Under Threat', '#FDFD11',
                    '#5A5AE2'
                ],
                'fill-opacity' : 0.6
            }
        })
```

A screenshot and the code for the park status colours.

Year Opened




```
map.addLayer({
            id : "park_shapes_layer",
            type : "fill",
            source : "parkShapesSource",
            paint : {
                'fill-color' : [
                    'match',
                    ['get', 'period_opened'],
                    '<1850', '#870E8A',
                    '1850-1874', '#CA1CCD',
                    '1875-1899', '#FC51FF',
                    '1900-1924', '#102C8F',
                    '1925-1949', '#244BD6',
                    '1950-1974', '#809CFF',
                    '1975-1999', '#BDCCFF',
                    '2000-2025', '#AAA30D',
                    '#000000'
                ],
                'fill-opacity' : 0.5
            }
        })
```

A screenshot and the code for the year opened colours.

Alterations

Screenshots of the alterations shapes and colours.

```
map.addLayer({
      id : "shrunk_park_areas",
      type : "fill",
      source : "parksAlterationsSource",
      paint : shrunkStyle, //This is a global variable at the top of the page defining the style.//
      filter : ["==", ["get", "alteration"], "Shrunk"]
})
map.addLayer({
    id : "park_alterations",
    type : "fill",
    source : "parkShapesSource",
    paint : {
        "fill-color" : [
            "match",
            ["get", "altered"],
            "Unchanged", "#50b848",
            "Expanded", "#059FDC",
            "Shrunk", "#50b848",
            "#000000"
        ],
        "fill-opacity" : 0.8
    }
})
map.addLayer({
    id : "expanded_park_areas",
    type : "fill",
    source : "parksAlterationsSource",
    paint : expandedStyle, //This is a global variable at the top of the page defining the style.//
    filter : ["==", ["get", "alteration"], "Expanded"]
})
```

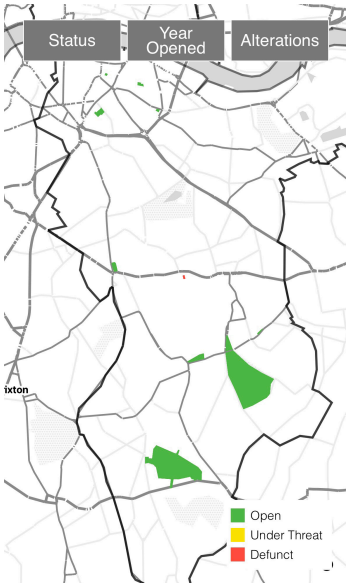
Screenshot (taken from Notion) of the code snippet I used to create the colours for the alterations.

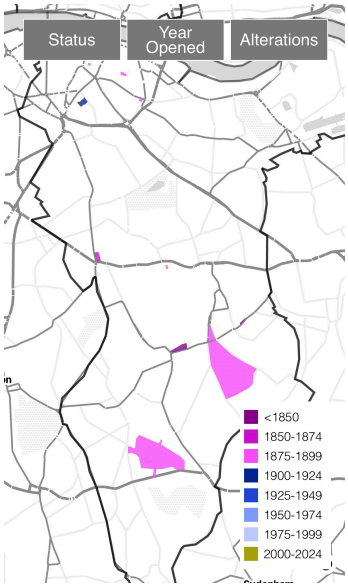
View Mode Buttons

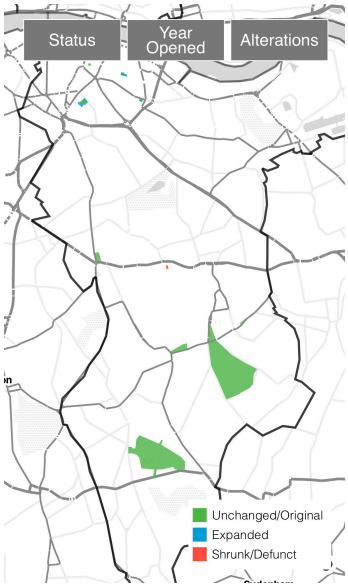
```
document.getElementById("status_button").addEventListener("click", function(){
    map.setPaintProperty("year_opened_layer", "fill-opacity", 0)
    map.setPaintProperty("park_status_layer", "fill-opacity", 0.8)
    map.setPaintProperty("shrunk_park_areas", "fill-opacity", 0)
    map.setPaintProperty("expanded_park_areas", "fill-opacity", 0)
    map.setPaintProperty("park_alterations", "fill-opacity", 0)
});
document.getElementById("year_opened_button").addEventListener("click", function(){
    map.setPaintProperty("year_opened_layer", "fill-opacity", 0.8)
    map.setPaintProperty("park_status_layer", "fill-opacity", 0)
    map.setPaintProperty("shrunk_park_areas", "fill-opacity", 0)
    map.setPaintProperty("expanded_park_areas", "fill-opacity", 0)
    map.setPaintProperty("park_alterations", "fill-opacity", 0)
});
document.getElementById("alterations_button").addEventListener("click", function(){
    map.setPaintProperty("year_opened_layer", "fill-opacity", 0)
    map.setPaintProperty("park_status_layer", "fill-opacity", 0)
    map.setPaintProperty("shrunk_park_areas", "fill-opacity", 0.8)
    map.setPaintProperty("expanded_park_areas", "fill-opacity", 0.8)
    map.setPaintProperty("park_alterations", "fill-opacity", 0.8)
});
```

🛠️07.02.24

I have added map keys to each of the view modes.







To make it easier to change colours globally, I have assigned each key colour a global variable. Underneath each global variable I have created a function which calls to each key whose colours should match that variable and apply the colours this way. Below is an example of this:

```
//Status Polygon Colours//
var statusOpenColor = "#50b848";
document.getElementById("status_open").style = `background-color: ${statusOpenColor}`;
document.getElementById("alterations_unchanged").style = `background-color: ${statusOpenColor}`;
var statusDefunctColor = "#ff4c3f";
document.getElementById("status_defunct").style = `background-color: ${statusDefunctColor}`;
var statusUnderThreatColor = "#fee400";
document.getElementById("status_under_threat").style = `background-color: ${statusUnderThreatColor}`;
```

The colour variables are also applied to each map layer as well, see example below:

```
var parkStatusViewMode = {
    id : "park_status_layer",
    type : "fill",
    source : "parkShapesSource",
    paint : {
        'fill-color' : [
            'match',
            ['get', 'status'],
            'Open', statusOpenColor,
            'Defunct', statusDefunctColor,
            'Under Threat', statusUnderThreatColor,
            '#5A5AE2'
        ],
        'fill-opacity' : 1
    }
};
```

To create the keys I used HTML and CSS. Below is the HTML and CSS for the Status Keys.

```
<div id="status_keys">
            <div class="key_row">
                <div id="status_open" class="key_square"></div>
                <div class="key_description">Open</div>
            </div>
            <div class="key_row">
                <div id="status_under_threat" class="key_square"></div>
                <div class="key_description">Under Threat</div>
            </div>
            <div class="key_row">
                <div id="status_defunct" class="key_square"></div>
                <div class="key_description">Defunct</div>
            </div>
        </div>
```

CSS

```
.key_row{
    display: flex;
    flex-direction: row;
    justify-content: flex-start;
    align-items: center;
    margin: 0.1rem;
}
.key_square{
    z-index: 3;
    width: 1rem;
    height: 1rem;
    margin: 0.1rem;
    margin-left: 0.2rem;
    background-color:aqua;
}
.key_description{
    font-family: 'Helvetica', sans-serif;
    font-size: 0.9rem;
    font-weight:lighter;
    margin-left: 0.3rem;
}
#status_keys{
    z-index: 2;
    width: 7.5rem;
    height: 5rem;
    background-color: white;
    display: flex;
    flex-direction:column;
    justify-content: center;
    position: absolute;
    bottom: 1rem;
    right: 1rem;
}
```

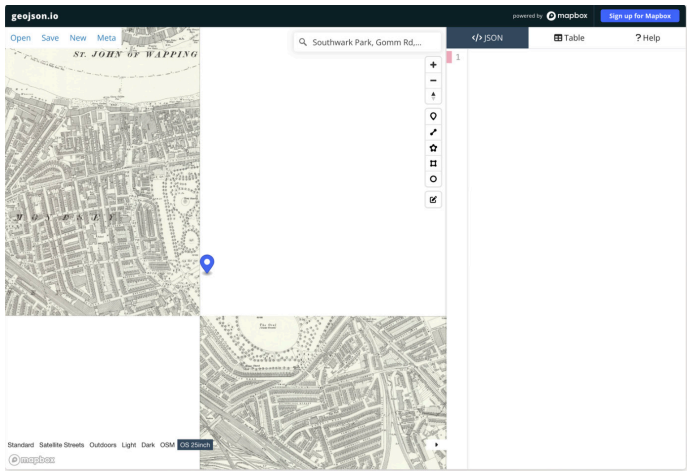
An example of the code for toggling between each key set depending on the view mode can be viewed below:

```
document.getElementById("status_button").addEventListener("click", function(){
    map.setPaintProperty("year_opened_layer", "fill-opacity", 0)
    map.setPaintProperty("park_status_layer", "fill-opacity", 0.8)
    map.setPaintProperty("shrunk_park_areas", "fill-opacity", 0)
    map.setPaintProperty("expanded_park_areas", "fill-opacity", 0)
    map.setPaintProperty("park_alterations", "fill-opacity", 0)

    document.getElementById("status_keys").style.display = "flex";
    document.getElementById("yearOpened_keys").style.display = "none";
    document.getElementById("alterations_keys").style.display = "none";
});
```

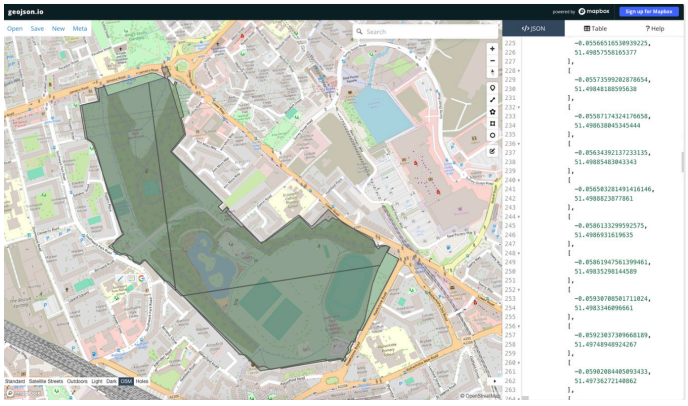
🛠️17.02.24

I have been having issues where when drawing out polygons in GeoJSON.io for altered parks, NLS's OS 25inch map layer has holes in it which appear on a separate layer which is made up of newer tiles. An example where this becomes a problem is Southwark Park who spans across multiple tile sets. The image below shows how part of Southwark park is missing due to the holes in the tile set.



Screenshot of geojson.io depicting how some of the old OS maps from National Library of Scotland's website have holes.

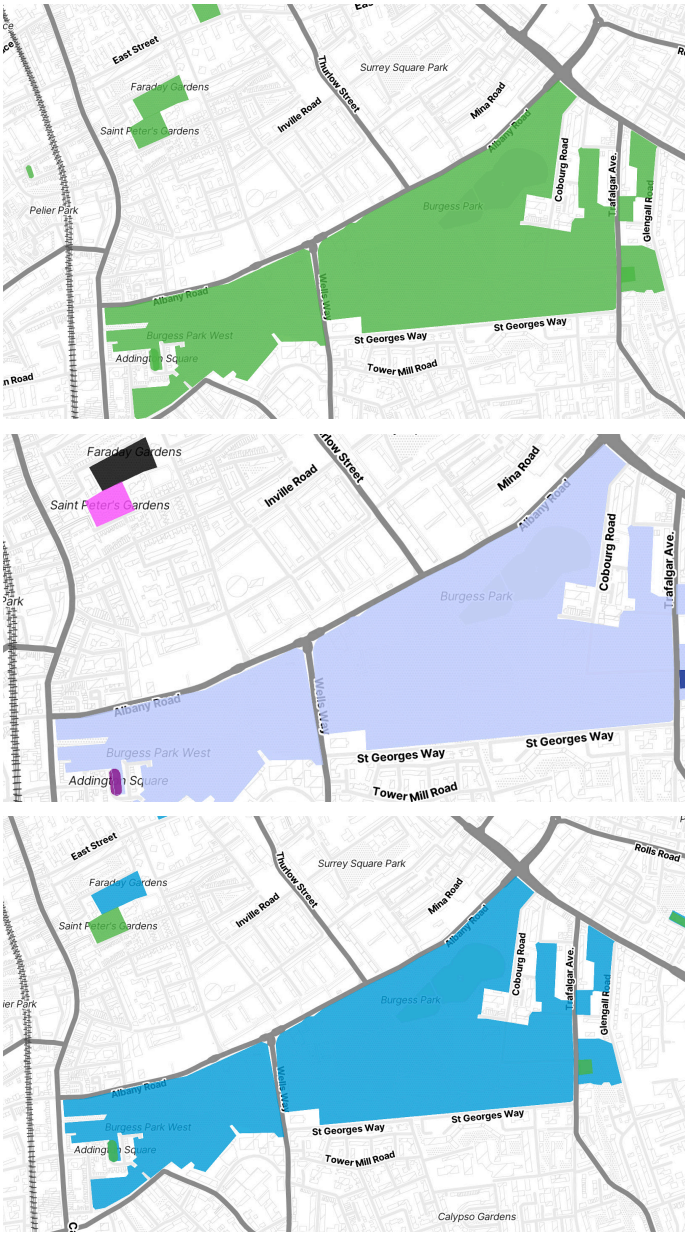
To resolve this problem I decided to draw polygons around the areas that are visible on the map. I then loaded in the tile set depicting the holes and then drew the areas of the park visible on that map. I then created another polygon this time of the whole park by tracing over the shapes I created using the two separate tile sets.



Screenshot of geojson.io depicting how I created multiple polygons to depict Southwark Park's shape accurately by using different base maps .

🛠️18.02.24

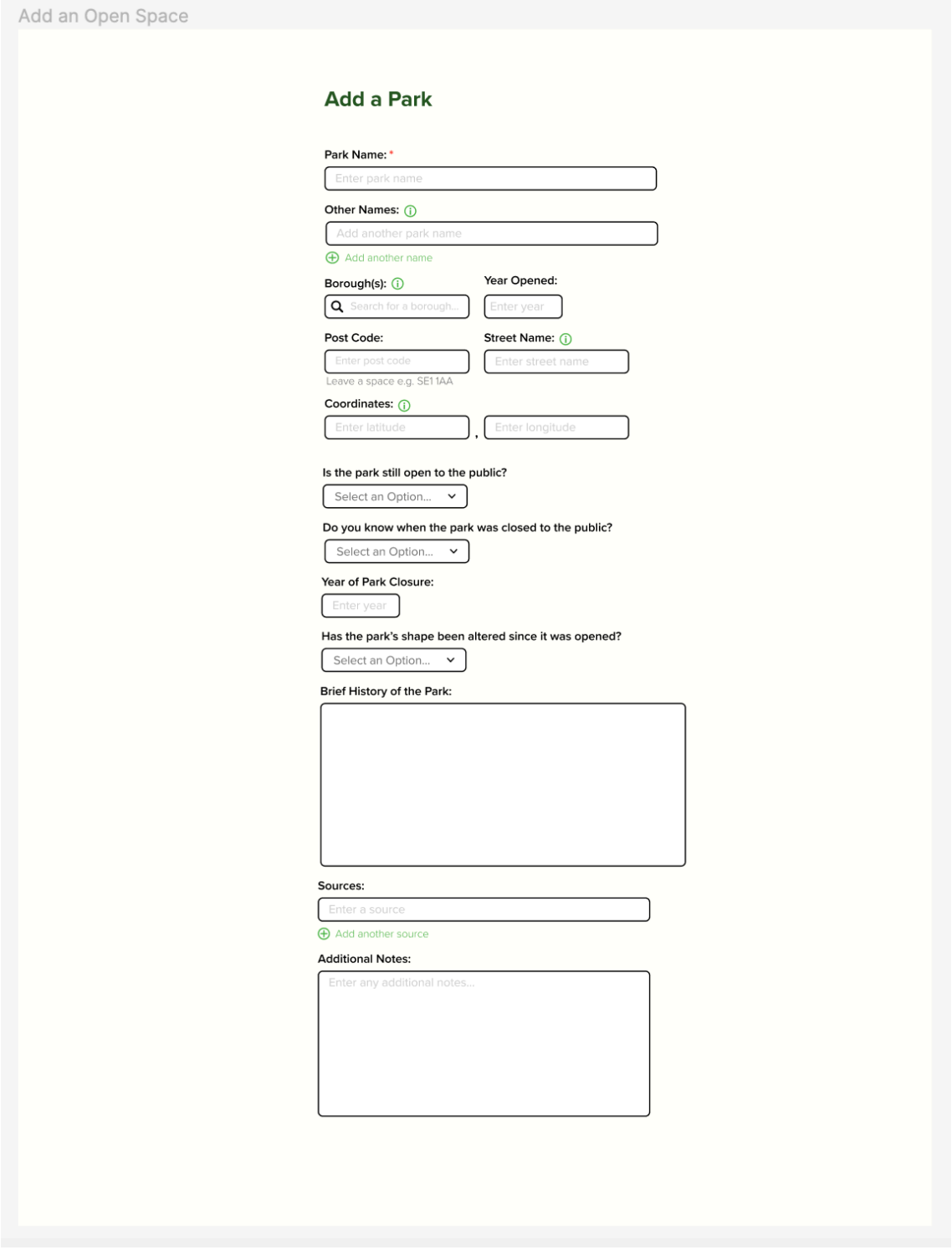
An issue I have been trying to resolve is around Burgess Park which is made up of many small parks that were opened earlier which were consolidated into one big park. For the ‘status’ view mode there isn’t any issue but for the ‘Year Opened’ and ‘Alterations’ panels I wanted to show the original areas, with their names showing. For the time being the solution I have come up with is simply to add those areas to both the park shapes list and park alterations lists so they appear as if they were any other park. In future I might try to streamline this further as when viewed in the status mode you can see the shapes of the original areas which don’t actually align with the current park boundaries. Also due to the opacity not being 100% the areas where the original park shape and Burgess Park shapes overlap are darker which looks odd. Also if the areas where the older areas are are clicked in the Status mode it will pop up with the original park names which may be confusing to users so I might remove this, although on the other hand it could be nice to see where the older areas are on the map. We shall see...



Screenshots of Burgess Park in various view modes.(From top to bottom: status view mode, year opened view mode and alterations view mode. You can see the older areas highlighted in different colours on the last two images.

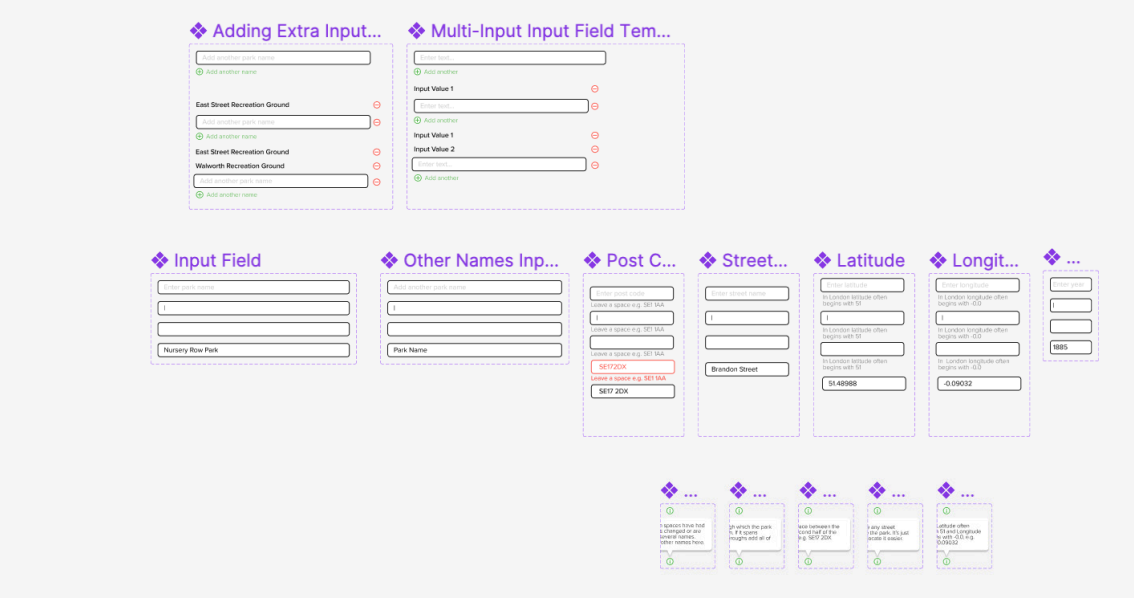
26.02-01.03.24

I have been taking a break from coding to focus on designing the UI of the website. During this week I have focused on designing a form for adding parks to the website. This will be aimed either at end users so anyone can submit information on parks or an administrative user. While I have been designing this interface while doing it I have decided to postpone building this functionality as it will be time consuming and isn't necessary to the basic functionality of the website. But the designs will be useful in demonstrating how this sort of functionality could be useful if the Parkive were deployed as a working product.



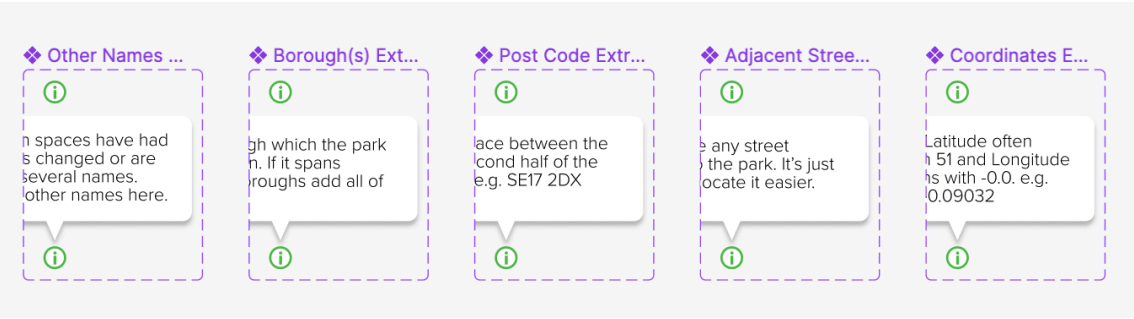
Screenshot of park information input form on Figma.

While creating the designs I also made fully functional components for each of the sections. This lead to the page looking very messy with components and variants scattered across the page. The screenshot below shows some of the mess, though by the time I remembered to take a screenshot I had already organised most of the components on a separate page, so imagine the mess but x3.



Figma components that have been organised and grouped together on a seperate page to the form designs.

I also realised, while creating the prototype that I can actually make one template of a component and edit it in each use case. For example below you can see five different information icons I created for different use cases. However I can actually get away we creating one default component and just change the text in each use case thus cutting down on the amount of clutter.



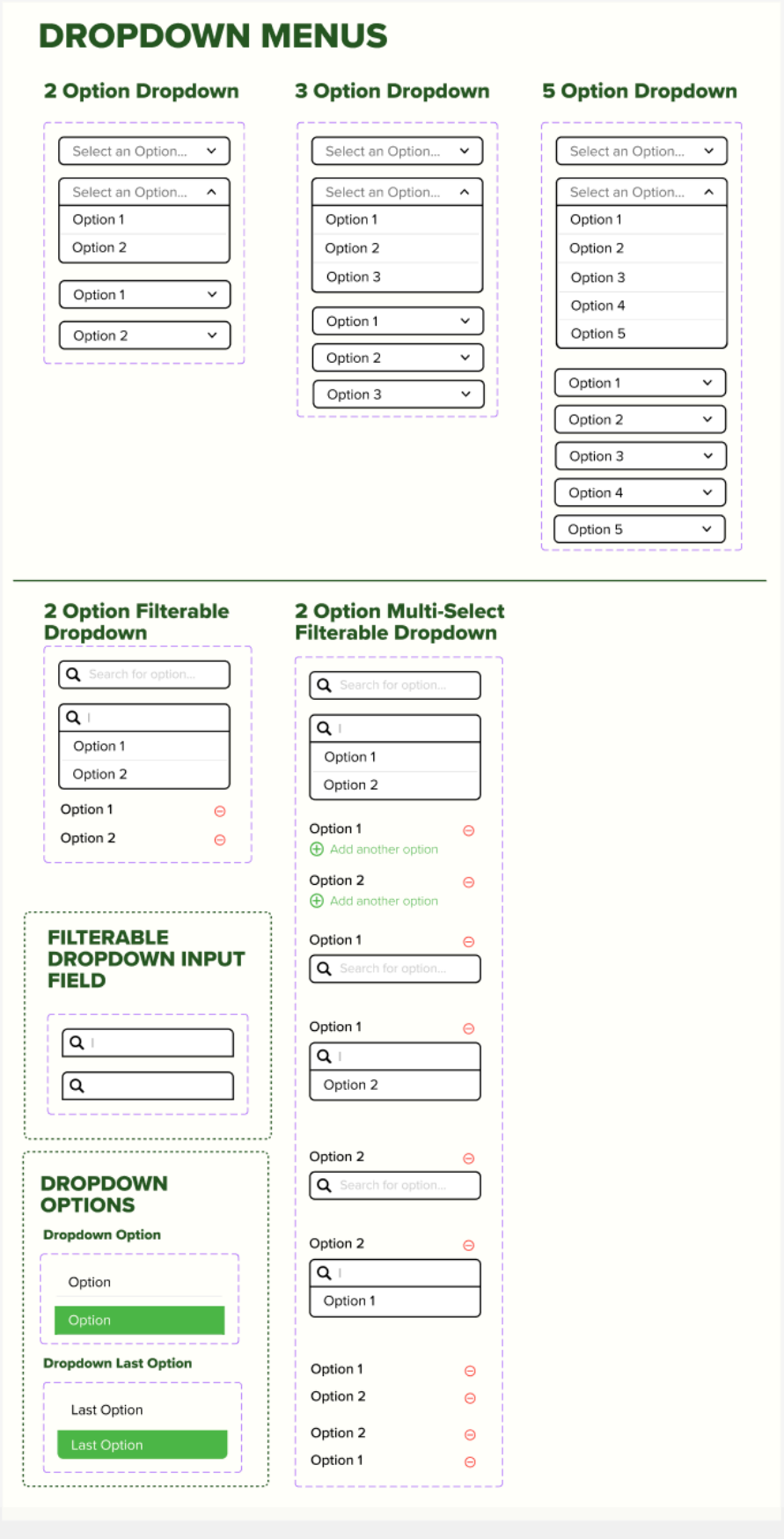
Screenshot depicting five different information icons and popups for each use case. I later discovered I could condense all of these into one component with different variants.

Kane showed me an example of a style sheet from his work on Figma and from this I got inspiration for how I might go about creating an organised style sheet with all the various components as well as fonts and colour schemes etc. I haven't finished it yet but below you can see where I have gotten up to.

By creating templates rather than a component for each use case it will save time in the future. These templates could also easily be used for future projects so will be worth keeping for that too.

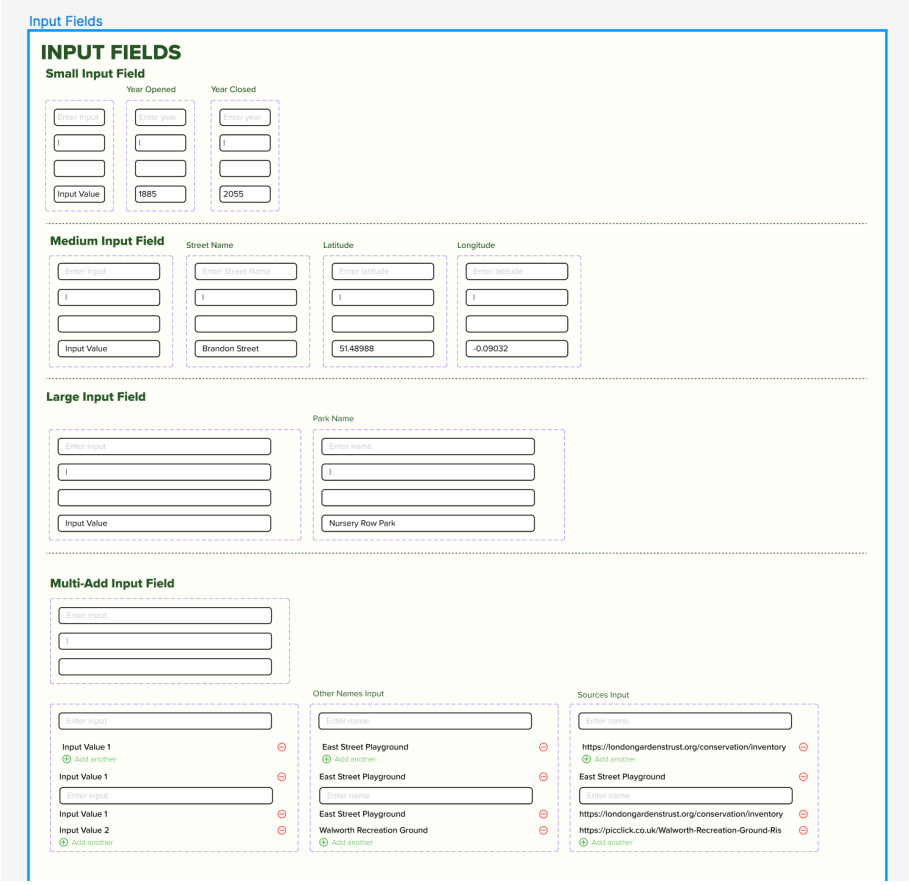
FORM STYLE SHEET

Dropdown Menus



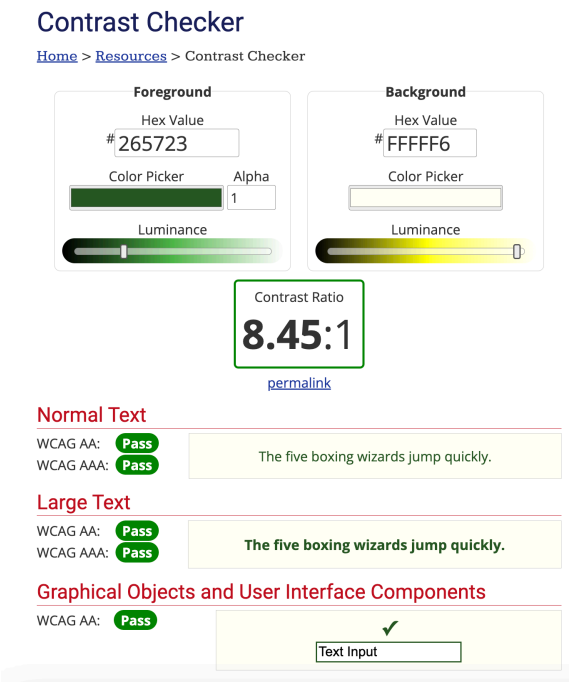
🎨04.-06.03.24

I finished making templates for each of the components and adding them to a separate page only to discover that actually I think I did need to create a unique version for each use case. Stupidly I deleted all of the unique components so I've been recreating variables of each of the components for each use case. The screenshot below shows all of the various variables for each of the components and how I have organised them.



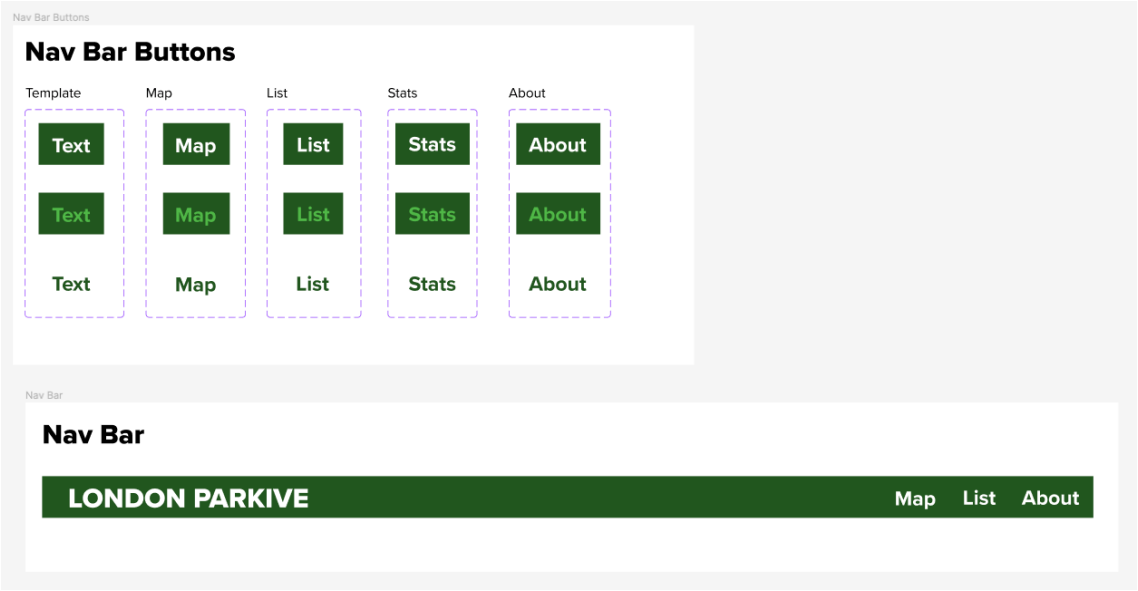
🎨07.03.24

Finally started on designing the actual Parkive prototype. I am trying to do this in as organised fashion as possible and as a result I didn't get a whole lot done today. However I feel like I am slowly getting the hang of using Figma in an efficient way it is just taking time to get used to it. Today I started to create a global style sheet including colour pallets, global text and components for the Navigation bar. I learned how to make the colour swatches and text global which will come in handy down the line. See screenshots of some of the work I have done below. In terms of decision making around the design the first nav bar I created felt too big so I shrank it a little and I think now it is a good size for a nav bar. Also I read up a little about good practices in UI and came across a contrast checker that will check if your colour scheme is accessible and my colour palette passed! Although I did later decide to use pure white (FFFFFF) as opposed to FFFFF6. This was because when paired with the base map which has a pure white background the off white looked a little odd.



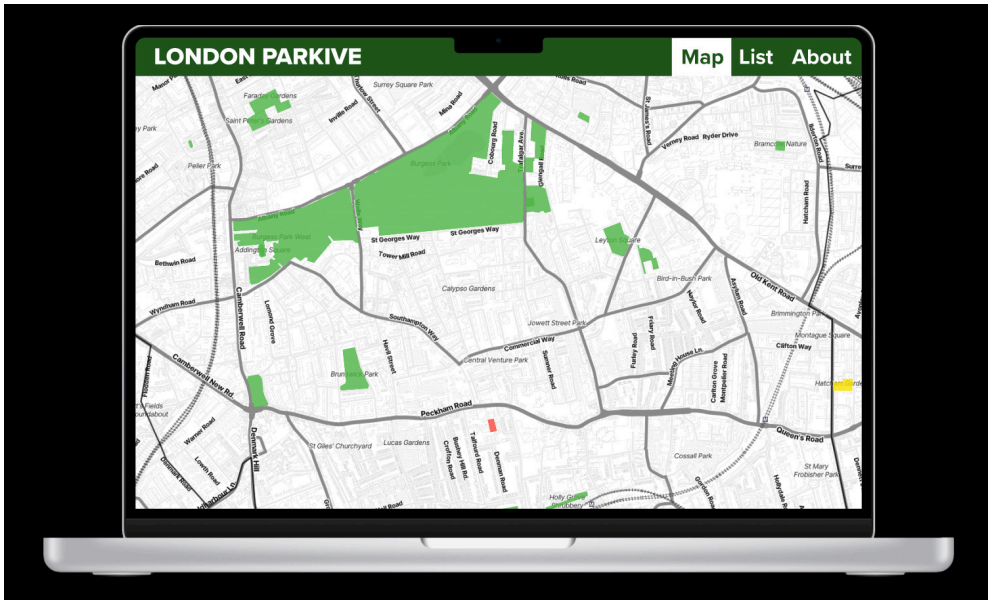
Screenshot of webaim.org's contrast checker. The colour scheme passed the test!

Colour swatches and global text I created on the new London Parkive Style Guide.



Nav Bar Buttons and Nav Bar, components I created for the Parkive on the style guide.

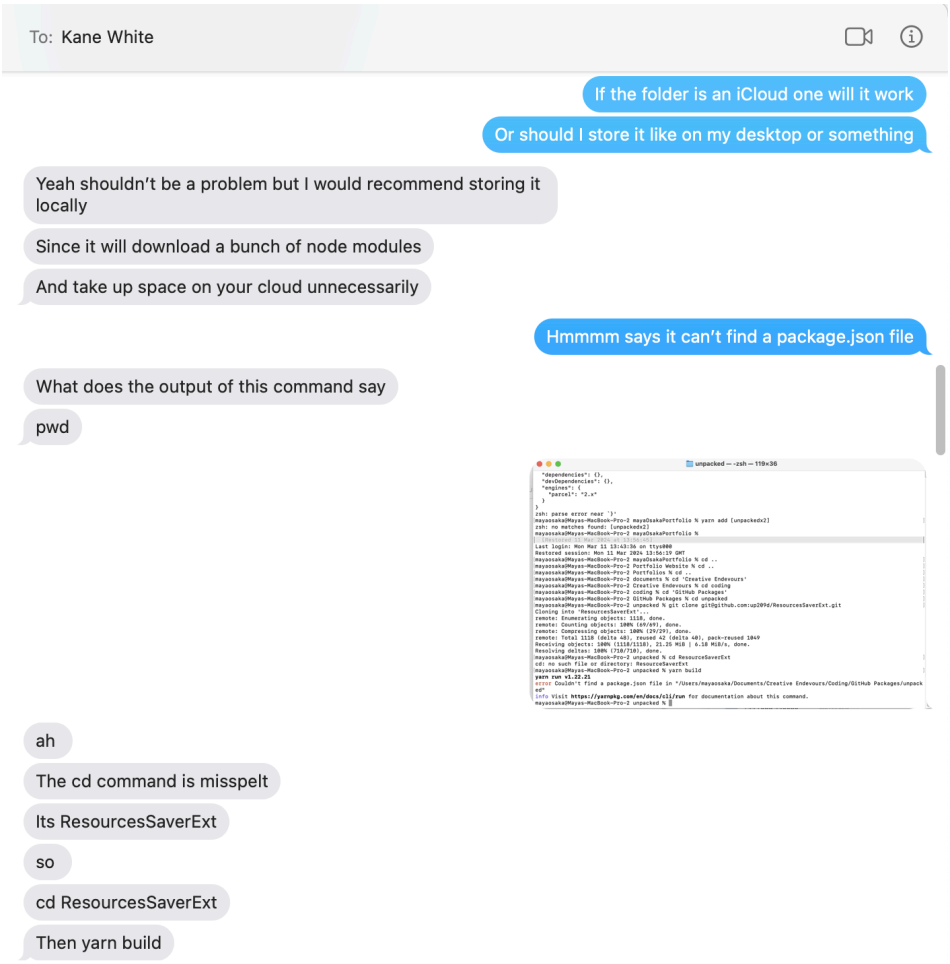
Original London Parkive nav bar. I felt like it was high and the text too big.



Modified nav bar.

🛠️11.03.24

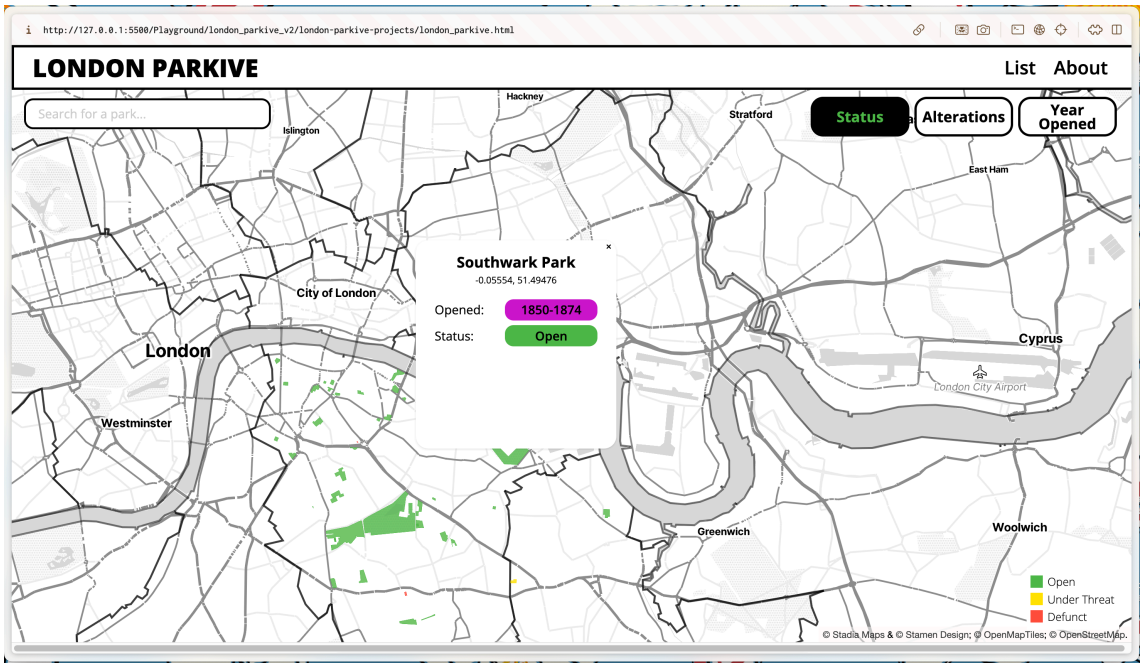
Started on creating geoJSON shapes for the remaining parks (of which there are roughly 90). It occurred to me over the weekend that, the bulk of these parks were created post 1951 which is the latest map available on National Library of Scotland's website. As the Southwark Councils website doesn't have APIs for their tiles I was starting to think about how I could draw the geoJSON shapes from more recent parks. One option that Kane suggested was to try downloading the individual PNGs of the map tiles from the website and try to piece them together myself and build a tool for drawing shapes. This all sounds complicated but the tool building part should have been quite simple as the documentation for how to achieve this can be found on MapBox and MapLibre. However downloading the PNGs proved tedious. First of all you can't download folders of images using the inspect window so this lead to me going down a rabbit hole trying to download this chrome extension that someone had built using github, the terminal and yarn (which I had never heard of before today). I proved incapable of figuring out how to do this on my own so relied heavily on Kane to help me get it up and running. In the end the tool didn't help as the tiles that were downloaded were only those present on the screen not all of the tiles I would need. After all this I discovered that actually I can draw shapes on Southwark Councils website and in fact it has proved a more functional tool than geoJSON.io which I had previously used. This is great because now I can do all of the shape drawing on this website instead. The only catch is having to convert from KSL format to geoJSON and reorder the geoJSON which is in the opposite order to how I like it (how I like it = properties on top, coordinates on bottom) but that is a fairly straight forward task. The discovery of **Option+Shift+F** which automatically optimises the code structure has been a massive time saver on all of this!



Screenshot of me and Kane's convosation where he helped me figure out how to get the chrome extension up and running.

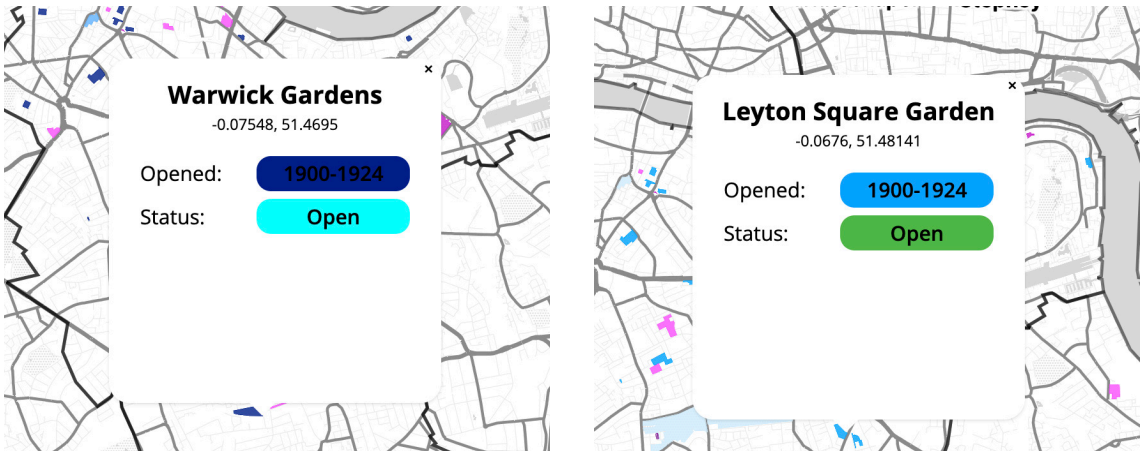
🛠️🎨12.03.24

I started the day by continuing to create GeoJSON shapes then Kane reminded me that actually this can be done at the end once I have finished building the website so I snapped out of it and continued building the actual website! I actually got quite a lot done today. I managed to create the nav bar, style the view mode buttons, added a search bar (though this doesn't function yet) and started styling the popups!



Screenshot of London Parkive website with newly added nav bar, search bar and popup. The view mode buttons have also been styled!

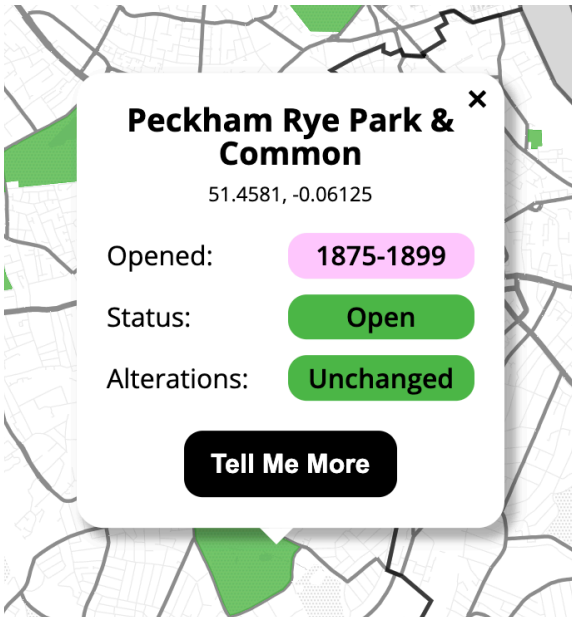
Along the way some design decisions were also made. I realised that some of the colours for the years opened are too dark to pair with a black font so I adjusted the blue toned colours so that they can be more visible with a black font. I also decided to take on George's advice from ages ago and make the range of blue the same blue but lighter opacity. I might change the colours again as it might be too difficult to distinguish the colours but for now it serves its purpose.



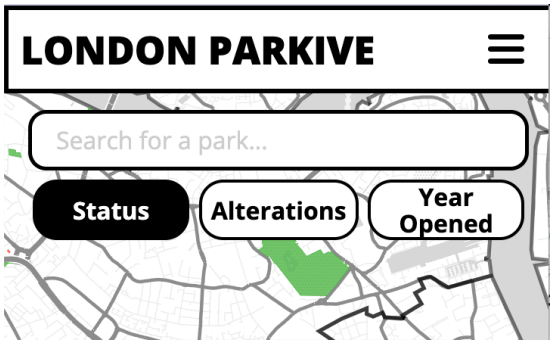
The screenshot on the left depicts a popup with the old 'period opened' colour scheme which is difficult to read. The screenshot on the right depicts the updated colour scheme.

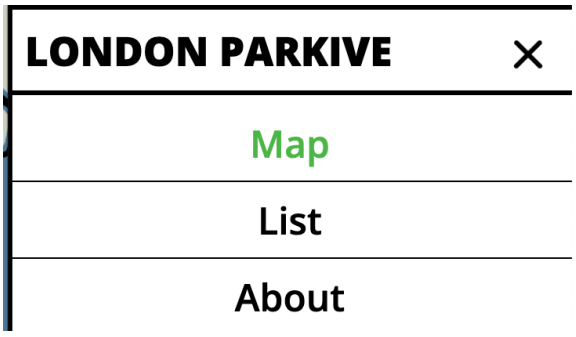
13-14.03.24

Over the past couple of days I have continued to build the website. Yesterday I finished making the popups. I am very pleased with the outcome because it looks almost identical to the Figma prototypes I created!



I then started working on creating a burger menu for the navigation bar so that when the viewport's width becomes smaller than 700px the navigation options will be stored in a dropdown menu rather than residing along the top of the page.

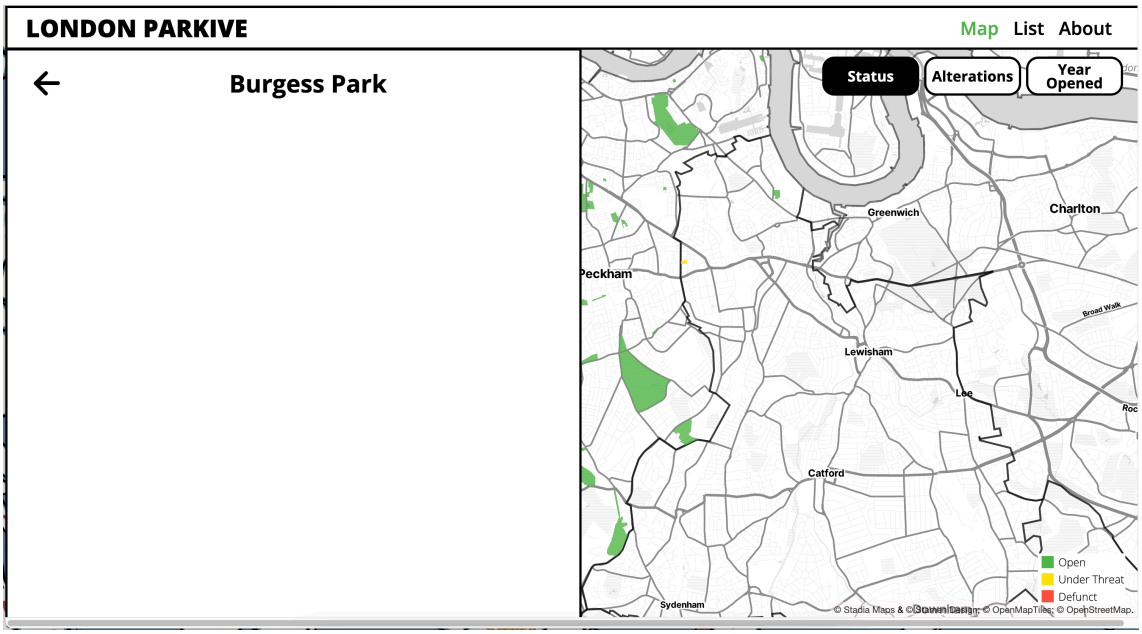





```
//Dropdown Menu for Mobile//
function myFunction(x) {
    let dropdownBackground = document.getElementById("dropdown_background");
    if (dropdownBackground.style.display === "flex"){
        dropdownBackground.style.display = "none";
    }
    else{
        dropdownBackground.style.display = "flex";
    };
    let navOptions = document.getElementById("nav_container");
    if(navOptions.style.display === "flex"){
        navOptions.style.display = "none";
    }
    else{
        navOptions.style.display = "flex";
    }
    x.classList.toggle("change");
};
```

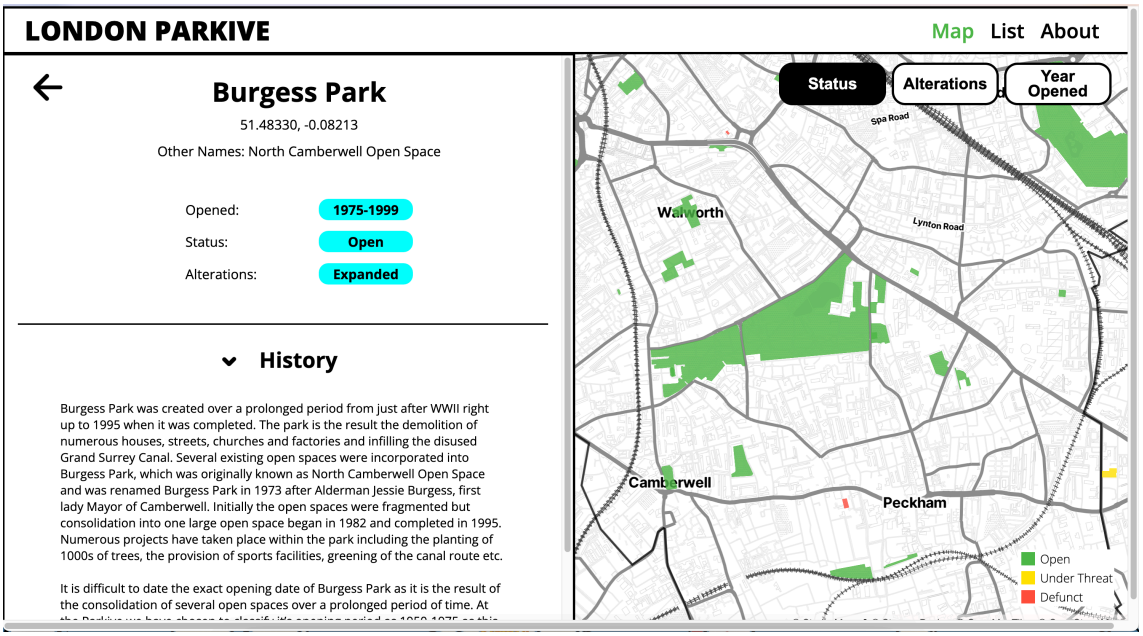
Once the burger menu became fully functional I spent some time working on the media queries to ensure that the Parkive can be viewed on all screen sizes.

After all this I made a start on creating the information overlay that will appear when the 'Tell me More' button is pressed.



🛠18.03.24

Continued adding to the overlay section. Managed to finish the header section consisting of the park name, other names, coordinates and overview info. Started on the more detailed sections of the overlay starting with the history section. Created an icon for the toggle.



🛠️19.03.24

Managed to get loads of stuff done today! (This is gonna be one chunky entry...)
I have taken several screen recordings but alas, I can't upload videos on ho a physical document BUT you can view it on the video reel!

- To begin with the "Tell Me More" button is now linked with the overlay so that when it is clicked the overlay pops up. Also when the back button is clicked the overlay disappears.
- The contents of the overlay is now dynamic and changes according to the park that has been clicked on.

This was achieved by creating a variable targeting the relevant property in the JSON file.

```
let clickedParkName = clickedFeatures[0].properties.name;
```

Then targeting the relevant HTML and changing the text:

```
document.getElementById("park_info_overlay_park_name").innerHTML = clickedParkName;
```

- The toggle arrow button now works so when it is clicked the history text appears and if it is clicked again the text is hidden.

```
function myFunction(x){
    let overlayHistoryText = document.getElementById("park_info_overlay_history_text");
    if (overlayHistoryText.style.display === "flex"){
        overlayHistoryText.style.display = "none";
    }
    else{
        overlayHistoryText.style.display = "flex";
    }
    x.classList.toggle("changeHistoryToggle");
}
```

- I made the arrow using CSS yesterday but made the active state today. It was difficult wrapping my head around the translate properties until I physically wrote down on a piece of paper that the first number is moving the object to the right and the second number is moving the object down.

```
.park_info_overlay_toggle_arrow_line_1{
    transform: translate(1px, 10.5px) rotate(45deg);
}
```

- When the "Tell Me More" button is clicked the popup disappears. The function for that is:

```
popup.remove();
```

- When the “Tell Me More” button is clicked the map zooms in on the clicked object. The basic function for that is:

```
map.flyTo({
            center: clickedParkCoordinates,
            zoom: zoomLevel
        })
```

Due to the variation in park sizes this was quite a complicated process. The flyTo function centers the map but half of the map is cut off by the overlay, so I had to add a discrepancy so that it would be centred in the half that is still depicted. The code below shows the center point with the added discrepancy.

```
let clickedParkCoordinates =
{lng: clickedFeatures[0].properties.longitude + longitudeDiscrepancy, lat: clickedFeatures[0].properties.latitude};
```

This helped but it soon became clear that due to the variation in sizes of parks a blanket discrepancy wouldn’t do. The screenshot below shows how when the discrepancy is calibrated towards a bigger park a smaller park would not end up being centered.

To fix this issue I decided to calculate the area of each park and create an if else statement which would determine how much discrepancy needed to be added depending on the size of the park. To achieve this I used Turf, a javaScript library for geo-spacial analysis. Turf has a function that passes in a geoJSON polygon and returns the area of that shape in square meters.

```
let clickedPark_sq_Meters = turf.area(clickedFeatures[0]);
```

I then converted it from square meters to acres as this seems to be a more common unit of measurement for parks. The conversion also rounds the number to the second decimal point.

```
function sqMetersToAcresConversion(squareMeters){
    clickedPark_acres = squareMeters / 4046.85642;
    rounded_area = Math.round(clickedPark_acres * 100) / 100;
    return rounded_area;
}
```

The next step was to create a function with an if else statement which would assign different discrepancies for different sized parks. In order to work out the right value for the discrepancy I clicked on different sized parks with the console open so I could see how big each park was (having console.logged the clickedParkArea) and assigned different values until I worked out which values worked best.

```
function determineLongitudeDiscrepancy(parkArea){
    if(parkArea >= 0 && parkArea < 1){
        return -0.0008;
    }
    else if(parkArea >= 1 && parkArea < 4){
        return -0.002;
    }
    else if(parkArea >= 4 && parkArea < 25){
        return -0.004;
    }
    else if(parkArea >= 25 && parkArea < 75){
        return -0.007;
    }
    else{
        return -0.015;
    }
}
```

The same process was repeated for the zoom level.

```
function determineParkZoom(parkArea){

    if(parkArea >=0 && parkArea < 1){
       return 18;
    }
    else if(parkArea >= 1 && parkArea < 2.5){
       return 17;
    }
    else if(parkArea >= 2.5 && parkArea < 4){
       return 16.5;
    }
    else if(parkArea >= 4 && parkArea < 10){
       return 16;
    }
    else if(parkArea >=10 && parkArea < 25){
       return 15.5;
    }
    else if(parkArea >=25 && parkArea < 75){
       return 15;
    }
    else{
        return 14;
    };
};
```

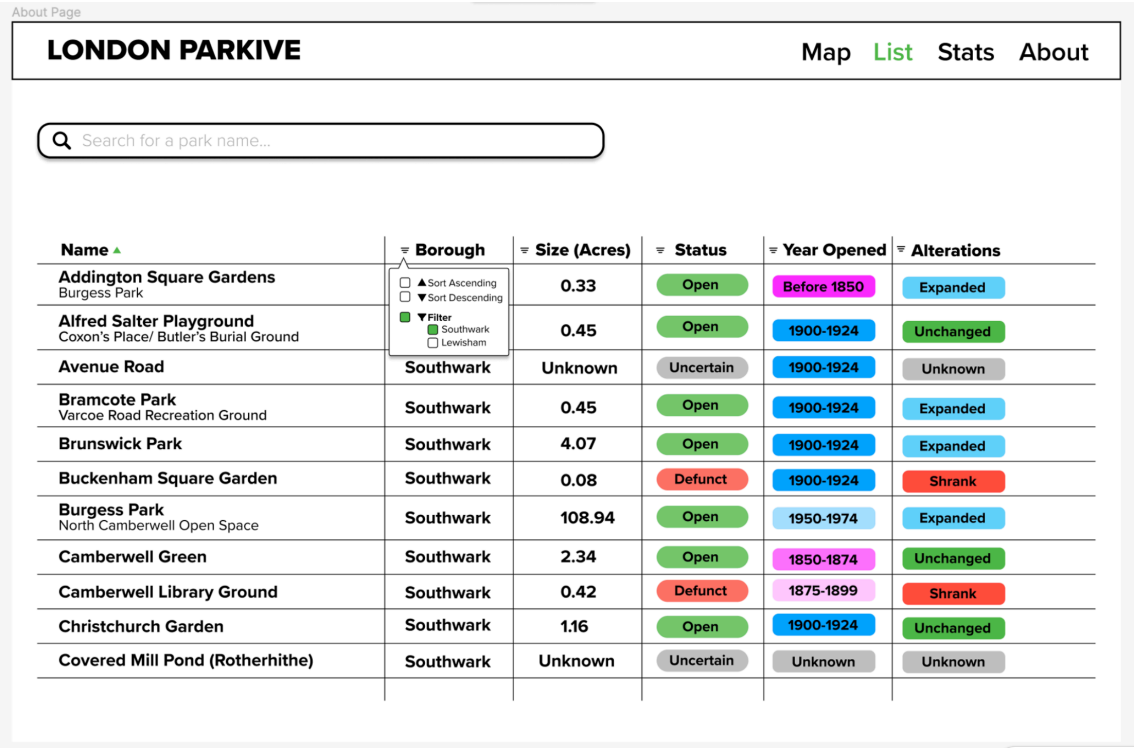
🛠️🎨21.03.24

- The first problem that I solved today was the fact that some of the parks (Burgess Park and Peckham Rye Park) have multiple polygons making up the park. With the area calculation function that I made a couple of days ago it would be very difficult to add the multiple areas together. There was also the issue that the area is only generated when a park shape is clicked on. To solve both issues I created a function that would work its way through the JSON file with all the polygons stored in it and work out the area of each park. Then it would store the area of the park in a new property. This would appear on the console and from there it can be copied and pasted into the JSON file replacing the original file that doesn't contain the new "Area" property.

```
function calculateGeoJSONPolygonArea(current_park){
        let polygon_square_meters = turf.area(current_park);
        let polygon_acres = polygon_square_meters / 4046.85642;
        let polygon_area = Math.round(polygon_acres * 100) / 100;
        current_park.properties.area = polygon_area;
        console.log(polygon_area);
    }
```

- After completing this I decided to focus my attention on the List page. What soon became apparent is that the javascript file I had created for the whole website doesn't work well for the list page because there is too much mapLibre related code on it. As the list page doesn't have a map or the corresponding ids and classes a whole series of error messages would come up. To fix this problem I decided to create a new javascript file for the pages that don't contain a map.

- Before continuing with building the list page I turned my attention to designing the page on Figma. It's funny how the better I get at programming the less patience I have with Figma. The screenshot below depicts the list view page I designed in Figma.I am now going to try and mimic this as closely as possible using code.

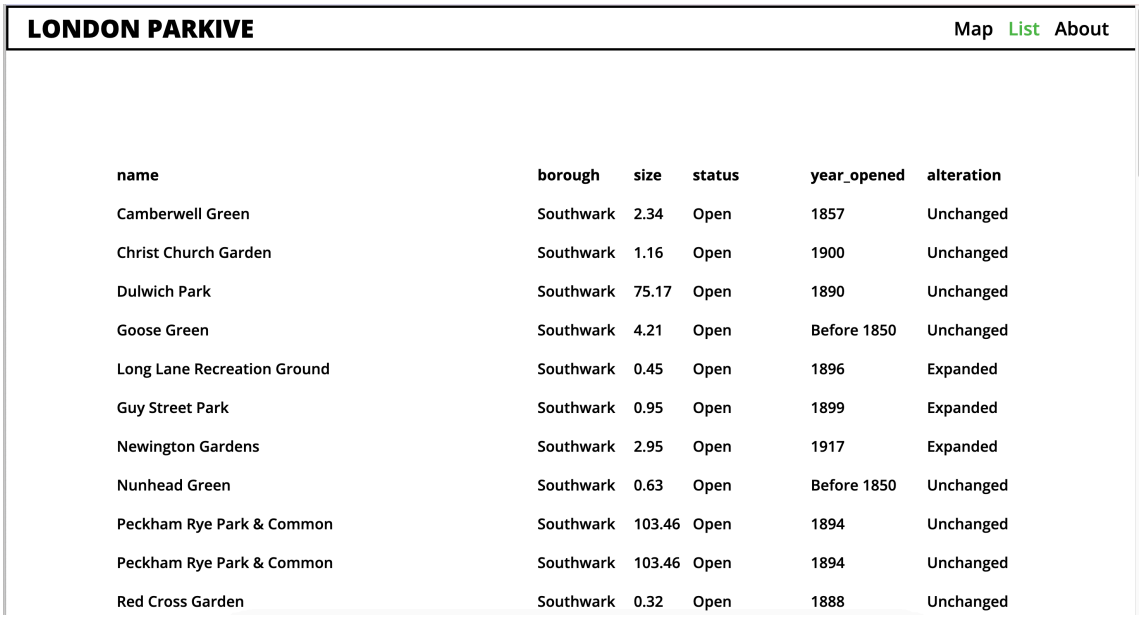


- The next step after this was starting to build the list. This was a challenge because I wanted to create a dynamic table that will automatically plug in the information from the JSON file into the table. To figure this out I asked chatGPT, whose methods worked! You can see my chat with chatGPT in the ChatGPT log.

```
function generateTable(data, propertiesToShow){
    const headerRow = document.getElementById('list_view_table_header');
   propertiesToShow.forEach(key =>{
        const th = document.createElement('th');
        th.textContent = key;
        headerRow.appendChild(th);
    });
    const tableBody = document.getElementById('list_view_table_body');
    data.features.forEach(feature => {
        const row = document.createElement('tr');
        propertiesToShow.forEach(key =>{
            const cell = document.createElement('td');
            cell.textContent = feature.properties[key];
            row.appendChild(cell);
        });
        tableBody.appendChild(row);
    });

};
fetch('./data_files/park_shapes_source.json')
    .then((response) => response.json())
    .then(response => {
        parksShapes = response;
        console.log(parksShapes);
        const tableProperties = ["name", "borough", "size", "status", "year_opened", "alteration"];
        generateTable(parksShapes, tableProperties);
    });
```

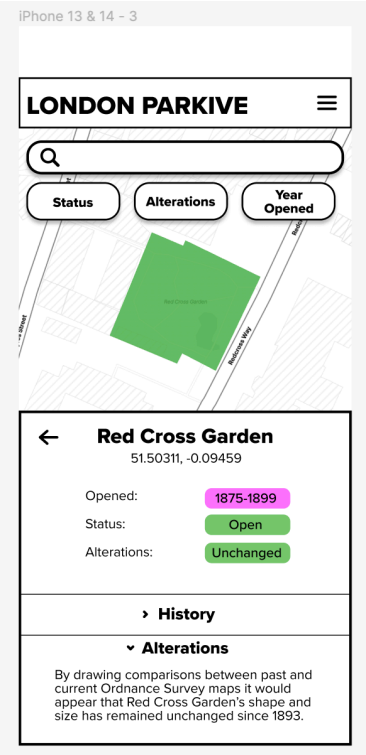
This is what the page currently looks like:



There is still a long way to go before the aesthetics match the Figma one 😄

🛠️🎨22.03.24

- Started on the media queries for the overlay view. In order to understand what needed to be done I made a mock up in Figma.



Translating this in to code was relatively straight forward apart from centring the parks. I had previously made a function to set the zoom level and add a discrepancy depending on the size of the park. I added to this function by adding another layer of if else statements that will change the zoom level and discrepancy based on the park size but also on the screen width.

```JavaScript
function determineLongitudeDiscrepancy(parkArea){
    if(window.innerWidth >= 1200){

        if(parkArea >= 0 && parkArea < 1){
            return -0.0008;
        }
        else if(parkArea >= 1 && parkArea < 4){
            return -0.002;
        }
        else if(parkArea >= 4 && parkArea < 25){
            return -0.004;
        }
        else if(parkArea >= 25 && parkArea < 75){
            return -0.007;
        }
        else{
            return -0.015;
        }
    }
```

😬 Code continues on to the next page...

```
    else if (window.innerWidth >= 900 && window.innerWidth < 1200){
        if(parkArea >= 0 && parkArea < 1){
            return -0.0008;
        }
        else if(parkArea >= 1 && parkArea < 4){
            return -0.002;
        }
        else if(parkArea >= 4 && parkArea < 25){
            return -0.004;
        }
        else if(parkArea >= 25 && parkArea < 75){
            return -0.006;
        }
        else{
            return -0.014;
        }
    }
    else{
        if(parkArea >= 0 && parkArea < 1){
            return -0.00008;
        }
        else if(parkArea >= 1 && parkArea < 4){
            return -0.0003;
        }
        else if(parkArea >= 4 && parkArea < 25){
            return -0.000006;
        }
        else if(parkArea >= 25 && parkArea < 75){
            return -0.00002;
        }
        else{
            return -0.001;
        }
    }
```

I also created another discrepancy for the latitude as well. I won't bother posting the code as it is more or less identical to the code pasted above. This is only required when the screen width is below 900px because at this point the overlay shifts from being on the left hand side of the screen to the bottom of the screen thus covering the bottom half of the screen.

🛠️🎨22.03.24

A bad bad day... I was trying to create a function where if an area outside of the overlay is clicked the overlay would close. I am yet to find a successful solution to this. Instead I have created a monster of a bug which I have been trying to fix all day without any luck... I managed to fix it briefly when I deleted all of the new code I added today but then it cropped back up again. I think this will be an issue for consultant Kane to help me figure out. The bug is that when the "Tell me More" button is clicked nothing popsup. In the console there is an error that the latitude is returning as not a number. But this only occurs when the screen size is larger than 900px. I have a feeling I know where the error is stemming from but cannot identify the error itself. I think it doesn't help that I am experiencing a mood slump, probably because I have been working all weekend (I normally take Sundays off) and am feeling a little burnt out. Hopefully with the assistance of consultant Kane tomorrow I can fix the problem.

```
Uncaught Error: Invalid LngLat object:          lng_lat.ts:63
(-0.07625, NaN)
    at new ec (lng_lat.ts:63:19)
    at ec.convert (lng_lat.ts:161:20)
    at Ba.Map.flyTo (camera.ts:1246:31)
    at HTMLButtonElement.<anonymous> (london_parkive.js:51
0:13)
```

🛠️😓26.03.24

Today has been a slow but productive day.

- First off, thanks to the help of Consultant Kane I was able to to fix the bug from yesterday. I feel I have got a lot to learn when it comes to the debugging process and using the Inspect tool effectively when debugging. The issue turned out to be fairly straight forward. This issue was in the function below:

```
function determineLatitudeDiscrepancy(parkArea){
    if(window.innerWidth >= 400 && window.innerWidth < 900){
        if(parkArea >= 0 && parkArea < 1){
            return -0.0006;
        }
        else if(parkArea >= 1 && parkArea < 4){
            return -0.0009;
        }
        else if(parkArea >= 4 && parkArea < 25){
            return -0.0028;
        }
        else if(parkArea >= 25 && parkArea < 75){
            return -0.005;
        }
        else{
            return -0.005;
        }
    }
    else if(window.innerWidth < 400) {
        if(parkArea >= 0 && parkArea < 1){
            return -0.0003;
        }
        else if(parkArea >= 1 && parkArea < 4){
            return -0.0004;
        }
        else if(parkArea >= 4 && parkArea < 25){
            return -0.0007;
        }
        else if(parkArea >= 25 && parkArea < 75){
            return -0.003;
        }
        else{
            return -0.003;
        }
    }
    else {
        return 0;
    };
};
```

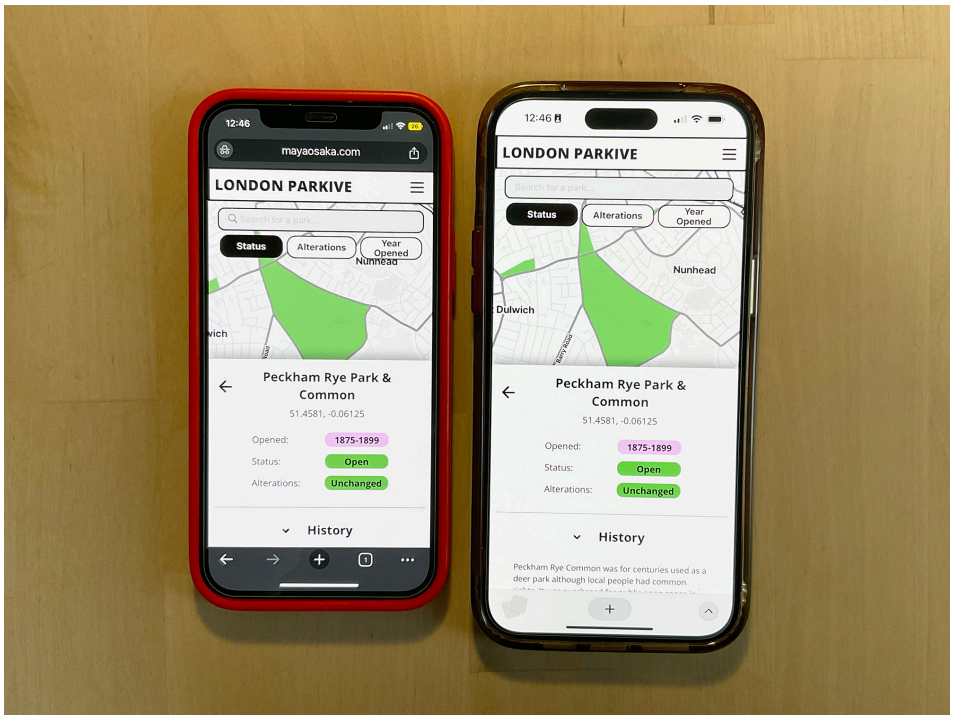
specifically, I hadn't added the 'else { return 0; };' at the bottom which meant that when the innerWidth was larger than 900 it would return a NaN. This is why the bug was only happening when the screen width was larger than 900.

- The next thing I achieved today was adding a function so when an area outside of the information overlay is clicked the overlay would disappear. This is the function I was originally trying to create yesterday. Again, Consultant Kane plaid a key role in figuring this out. I had asked chatGPT for a solution which was the correct solution but in the context that I was using it was lacking. The final solution that worked is on the next page (irrelevant code has been removed in this example):

```
    document.getElementById("tell_me_more_button").addEventListener("click",function(clickedFeature){
          clickedFeature.stopPropagation();
          mapContainer.addEventListener("click", closeOverlay);
      });
});
let parkInfoOverlay = document.getElementById("parks_information_overlay_container");
function closeOverlay(){
    if(parkInfoOverlay.style.display === "block"){
        parkInfoOverlay.style.display = "none";
    }
}
```

The key here is the **stopPropagation()** function which is something that I hadn't previously encountered before. Hence Consultant Kane saved the day by telling me about this function. I then went down a rabbit hole of looking into the **stopPropagation()** function and **Event Bubbling** and **Event Capturing** which I have written about in detail in the Useful Stuff section. I also wanted to add an element to the function that would make the map zoom out slightly when the overlay is closed to counter the fact it zooms in when the overlay opens. However due to already being slightly behind schedule I have decided to postpone doing this for now.

- Another thing I worked on today was continuing to develop the media queries for the zoom effect. I noticed that while on Kane's iPhone 14pro Max the parks center well when zoomed in, on my smaller iPhone 12 they do not centre as well. This took a lot of trial and error as did all the previous iterations for getting the centring right but it has paid off because now it centres well on my phone too.

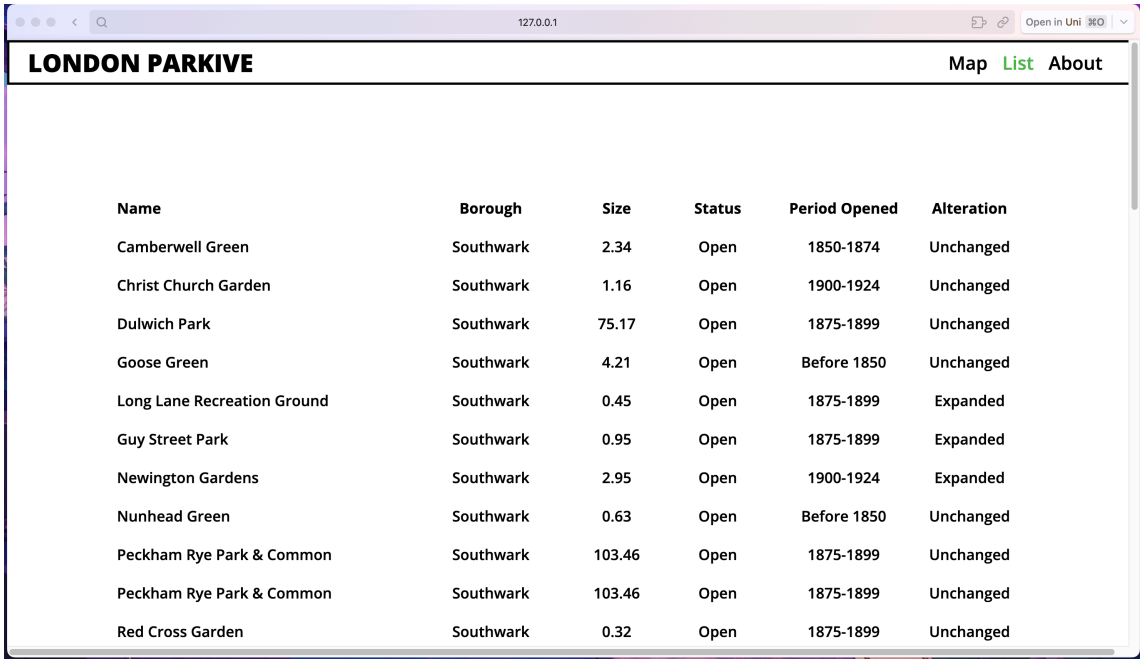


- The final thing I have worked on today is continuing working on the list page. I have worked on two key areas:

1. The header row of the table was initially just the key names from the JSON file. However the first letters of the names were lower case (e.g. size instead of Size) and multi-word titles had a _ between the words (e.g. year_opened instead of Year Opened). I turned to chatGPT to help me fix this. The solution I used is on the next page (I have just pasted in the relevant bits):

```
var headerMapping = {
      "name" : "Name",
      "borough" : "Borough",
      "size" : "Size",
      "status" : "Status",
      "year_opened" : "Period Opened",
      "alteration" : "Alteration"
  };
   function generateTable(data, propertiesToShow){
      const headerRow = document.getElementById('list_view_table_header');
      for (var key in headerMapping){
          var th = document.createElement("th");
          th.textContent = headerMapping[key];
          headerRow.appendChild(th);
      }
```

2. The final thing I did today was start styling the table but I haven't got very far with this at all so I will post more about it tomorrow. The image below shows what the list page looks like after today.



Name	Borough	Size	Status	Period Opened	Alteration
Camberwell Green	Southwark	2.34	Open	1850-1874	Unchanged
Christ Church Garden	Southwark	1.16	Open	1900-1924	Unchanged
Dulwich Park	Southwark	75.17	Open	1875-1899	Unchanged
Goose Green	Southwark	4.21	Open	Before 1850	Unchanged
Long Lane Recreation Ground	Southwark	0.45	Open	1875-1899	Expanded
Guy Street Park	Southwark	0.95	Open	1875-1899	Expanded
Newington Gardens	Southwark	2.95	Open	1900-1924	Expanded
Nunhead Green	Southwark	0.63	Open	Before 1850	Unchanged
Peckham Rye Park & Common	Southwark	103.46	Open	1875-1899	Unchanged
Peckham Rye Park & Common	Southwark	103.46	Open	1875-1899	Unchanged
Red Cross Garden	Southwark	0.32	Open	1875-1899	Unchanged

🛠️🤦.03.24

I have made good progress today 👉 Though I am starting to fall a little behind schedule 🤢 But I am trying not to worry because I am learning a ton which is great and the more I panic when I code the less I get done. Today I have focused on building the list page. I have:

Made aesthetic improvements by adding borders to selected edges to make the table look more like the Figma designs. Along the way I learned some new CSS styling!

- **`border-collapse`** is a style you can apply to tables so there aren't any gaps between the borders.
- **`table-layout: fixed;`** prevents the table from changing shape depending on the amount of content there is and fixes its positioning instead.
- **`#table_header th:nth-child(1){}`** is a method of naming allows you to target specific elements. In this case it is targeting the first child element of the table_header element.

For most of the day I was working out how to add divs into table cells that have been dynamically made (dynamically meaning instead of being inputed in the HTML file the elements have been created in javaScript with content that varies depending on property values etc.) and then styling the div element to look like the colour coded boxes on the popups on the map so the colours change depending on the values. The bit that I found particularly tricky was figuring out how to specifically target the columns that I wanted to add the colour coded boxes to, because the other columns don't require them. I tried several methods (stupidly forgot to make a note of them but I will describe what I was trying to attempt, it is also important to note that these methods might have worked if I had continued working on the method):

Attempt 1:
After generating a generic set of cells trying to specifically target the relevant cells with if else statements and add the divs in.

Attempt 2:
Adding a div to all the cells, setting the default as transparent and then targeting the relevant columns with conditional (if else) statements to add an additional class and colour code the divs using a switch statement.

Successful Attempt:
Created an if else statement targeting the relevant keys first then creating those cells with variables that are specific to each condition then creating a container for the coloured div to sit in (so the display can be set to flext so the div can be easily centered and the size adjusts more easily depentding on the screen width). Then within that div create the colour coded div which is named according to the properties it has. A switch statement embeded within the if else statement determines the background color of the box depending on the properties it has. You can see the code that worked below:

JavaScript Copy Cap

```
data.features.forEach(feature =>{
            const row = document.createElement('tr');
            propertiesToShow.forEach(key => {

                if(key === "status"){
                    const statusCell = document.createElement('td');
                    row.appendChild(statusCell);
                    var statusDivContainer = document.createElement("div");
                    statusDivContainer.classList.add("list_view_box_container")
                    statusCell.appendChild(statusDivContainer);
                    var statusDiv = document.createElement("div");
                    statusDiv.innerHTML = feature.properties[key];
                    statusDiv.classList.add("list_view_box");
                    switch(feature.properties[key]){
                        case "Open": statusDiv.style.backgroundColor = status_open_color;
                        break;
                        case "Defunct": statusDiv.style.backgroundColor = status_defunct_color;
                        break;
                        case "Under Threat": statusDiv.style.backgroundColor = status_under_threat_color;
                        break;
                        case "Unknown": statusDiv.style.backgroundColor = unknown_color;
                        default:
                        break;
                    }
                    statusDivContainer.appendChild(statusDiv);
                }
```

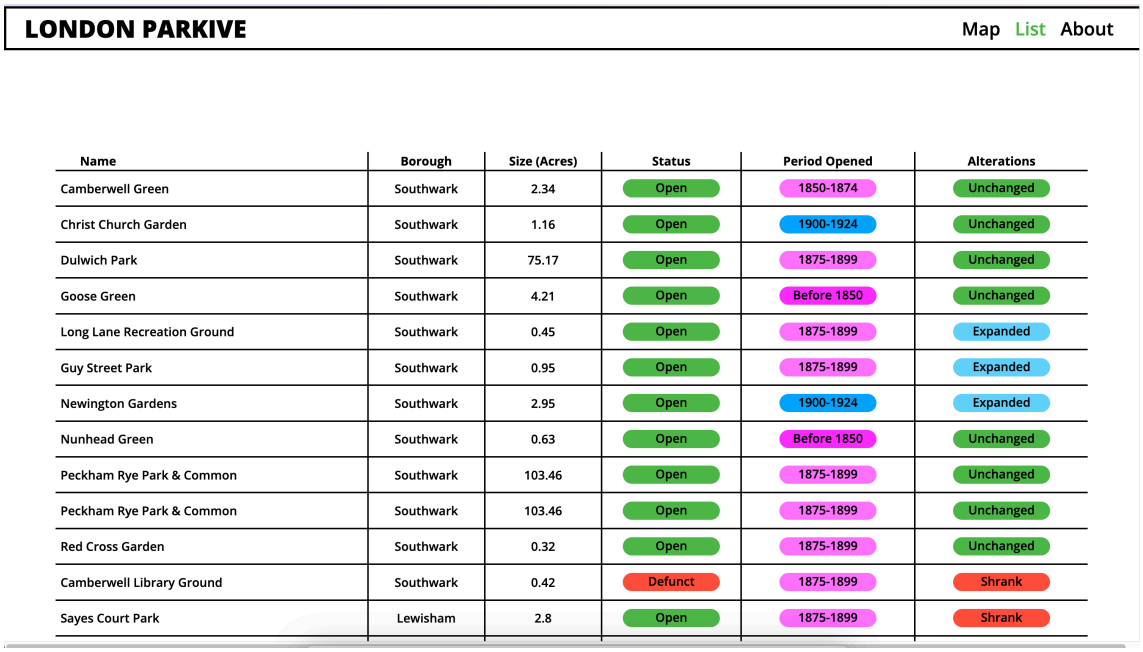
😬 Code continues on to the next page...

```
            else if(key === "period_opened"){
                const periodOpenedCell = document.createElement('td');
                row.appendChild(periodOpenedCell);
                var periodOpenedDivContainer = document.createElement("div");
                periodOpenedDivContainer.classList.add("list_view_box_container");
                periodOpenedCell.appendChild(periodOpenedDivContainer);
                var periodOpenedDiv = document.createElement("div");
                periodOpenedDiv.innerHTML = feature.properties[key];
                periodOpenedDiv.classList.add("list_view_box");
                switch(feature.properties[key]){
                    case "Before 1850": periodOpenedDiv.style.backgroundColor = yo_before1850_color;
                    break;
                    case "1850-1874": periodOpenedDiv.style.backgroundColor = yo_1850to1874_color;
                    break;
                    case "1875-1899": periodOpenedDiv.style.backgroundColor = yo_1850to1874_color;
                    break;
                    case "1900-1924": periodOpenedDiv.style.backgroundColor = yo_1900to1924_color;
                    break;
                    case "1925-1949": periodOpenedDiv.style.backgroundColor = yo_1925to1949_color;
                    break;
                    case "1950-1974": periodOpenedDiv.style.backgroundColor = yo_1950to1974_color;
                    break;
                    case "1975-1999": periodOpenedDiv.style.backgroundColor = yo_1975to1999_color;
                    break;
                    case "2000-2024": periodOpenedDiv.style.backgroundColor = yo_2000to2024_color;
                    break;
                    case "Unknown": periodOpenedDiv.style.backgroundColor = unknown_color;
                    break;
                    default: periodOpenedDiv.style.backgroundColor = "cyan";
                    break;
                }
                periodOpenedDivContainer.appendChild(periodOpenedDiv);
            }
            else if(key === "alteration"){
                const alterationsCell = document.createElement("td");
                row.appendChild(alterationsCell);
                var alterationsDivContainer = document.createElement("div");
                alterationsDivContainer.classList.add("list_view_box_container");
                alterationsCell.appendChild(alterationsDivContainer);
                var alterationsDiv = document.createElement("div");
                alterationsDiv.innerHTML = feature.properties[key];
                alterationsDiv.classList.add("list_view_box");
                switch(feature.properties[key]){
                    case "Unchanged": alterationsDiv.style.backgroundColor = status_open_color;
                    break;
                    case "Expanded": alterationsDiv.style.backgroundColor = alterations_expanded_color;
                    break;
                    case "Shrank": alterationsDiv.style.backgroundColor = status_defunct_color;
                    break;
                    case "Unknown": alterationsDiv.style.backgroundColor = unknown_color;
                }
                alterationsDivContainer.appendChild(alterationsDiv);
            }
            else{
                const cell = document.createElement('td');
                row.appendChild(cell);
                cell.innerHTML = feature.properties[key];
            }
```

I have learned some useful javaScript along the way. In particular:

- **createElement()** This function allows you to create new element in the DOM. For example **document.createElement("div")** would create a new div element.
- **appendChild()** This function allows you to assign an element to a parent element. For example if there was an existing div with the variable name currentDiv and you created a new div using createElement() which you named newDiv you could make currentDiv the parent element of newDiv by writing **currentDiv.appendChild(newDiv)**.
- **classList.add()** adds a class name to a div. For example if I wanted to add the class "new_div" to the newly created div element I could write **newDiv.classList.add("new_div")** and it would add a new class name to the div.
- **innerHTML** I have previously used this function but don't know if I've talked about it before. The innerHTML function (? I don't actually know if it technically a function) allows you to target the space between an opening and closing HTML tag and add text in it. For example, if we wanted to add the text "Hello" in the newDiv tag you could write **newDiv.innerHTML = "Hello"**

The end result of all of this can be seen below:



LONDON PARKIVE Map List About

Name	Borough	Size (Acres)	Status	Period Opened	Alterations
Camberwell Green	Southwark	2.34	Open	1850-1874	Unchanged
Christ Church Garden	Southwark	1.16	Open	1900-1924	Unchanged
Dulwich Park	Southwark	75.17	Open	1875-1899	Unchanged
Goose Green	Southwark	4.21	Open	Before 1850	Unchanged
Long Lane Recreation Ground	Southwark	0.45	Open	1875-1899	Expanded
Guy Street Park	Southwark	0.95	Open	1875-1899	Expanded
Newington Gardens	Southwark	2.95	Open	1900-1924	Expanded
Nunhead Green	Southwark	0.63	Open	Before 1850	Unchanged
Peckham Rye Park & Common	Southwark	103.46	Open	1875-1899	Unchanged
Peckham Rye Park & Common	Southwark	103.46	Open	1875-1899	Unchanged
Red Cross Garden	Southwark	0.32	Open	1875-1899	Unchanged
Camberwell Library Ground	Southwark	0.42	Defunct	1875-1899	Shrank
Sayes Court Park	Lewisham	2.8	Open	1875-1899	Shrank

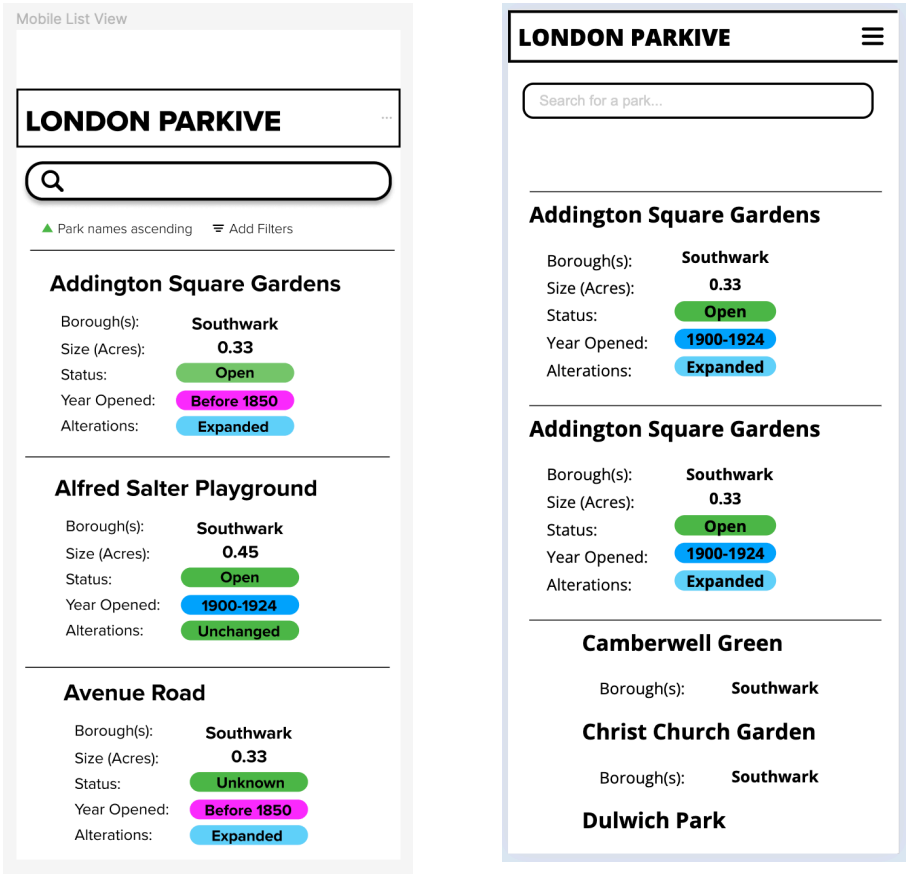
🛠️🎨28.03.24

Today has been a fairly productive day. I spent the day working on the Media Queries for the List View Page. I started off by doing a little research to find out what common solutions are when it comes to depicting tables on mobile. The article titled 'When you cannot run away from using tables on mobile' on UXDesign.cc came in particularly handy (the link is listed below).

https://uxdesign.cc/when-you-cannot-runaway-of-using-tables-on-mobile-630923bcea72

After doing some research I spent some time in Figma designing the mobile version of the list page.

I then started building it. The method I went for is to create a new series of divs for the mobile view and hide the table when the screen width drops below a certain size. The new divs, rather than being a table are just divs. The information from the JSON file is created in the same way that the table was created but using elements instead of <table> elements. The snippet of code below demonstrates what the code looks like.



The screenshot on the left is the figma mock up that I made. The screenshot on the right is of the London Parkive list view. The top sections [that are titled Addington Square Gardens] are made from static HTML elements so I could get a feel for what they should look like before dynamically generating the list.

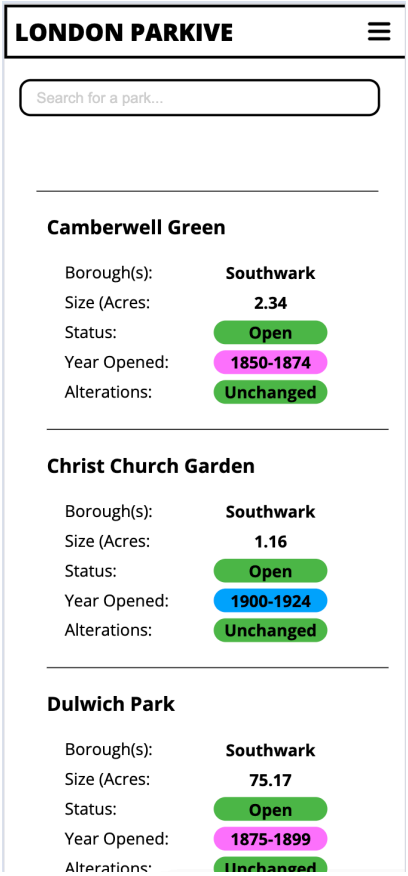
```
function generateMobileListView(data){
        var mobileListContainer = document.getElementById("mobile_table_container");
        data.features.forEach(feature =>{

            var mobileListParkSection = document.createElement("div");
            mobileListParkSection.classList.add("mobile_list_view_park_container");
            mobileListContainer.appendChild(mobileListParkSection);

            var mobileListParkName = document.createElement("div");
            mobileListParkName.classList.add("mobile_list_view_park_name");
            mobileListParkName.innerHTML = feature.properties.name;
            mobileListParkSection.appendChild(mobileListParkName);
```

🛠️29.03.24

Today I have finished the Media Queries for the mobile list view (Yay!)



For the remainder of the day I worked on the functionality of when a park name on the list is clicked and the user is transported the corresponding map overlay page. This is a rather complex task, or more accurately the sort of task that I haven't previously encountered so Consultant Kane sat with me and talked me through how to achieve this and along the way has helped me tidy up sections of my code. This is what we have covered today:

Added a click event listener to the park name element on the mobile list view mode which triggers a function which opens the map page.

The Event Listner

```
mobileListParkName.addEventListener("click", function kanesCoolClickHanlder(){
navigateToMapPage(feature.properties)
});
```

The function that opens the map page

```
function navigateToMapPage(park_data){
    console.log(park_data);
    window.localStorage.setItem("list_view_clicked_park", JSON.stringify(park_data));
    var baseurl = window.location.pathname;
    var spliturl = baseurl.split("lp_list_page.html");
    var targeturl = spliturl[0] + "london_parkive.html";
    console.log(targeturl);
    var targetHref = window.location.origin + targeturl;
    console.log(targetHref);
    window.open(targetHref, "_self");
}
```

I learned a few new things along the way:

- **Window** reprasents where DOM documents are stored and in a nut shell is a good thing to target if you want to achieve something on a global scope.
- **split()** allows you to split a string. The split string is stored in an array.
- Organising JavaScript code and better practice including:

1. Reusable Functions. This function, which opens the parks overlay, used to be em bedded in an event handler that was triggered when the “Tell Me More” button was clicked. The same function will be used when a park name on the list page is clicked so it makes sense for it to be reusable. This is achieved by creating an object containing the park data.

```
function openParkOverlay(clickedFeature, park_data){
    clickedFeature.stopPropagation();
    popup.remove();
    map.flyTo({
        center: park_data.clickedParkCoordinates,
        zoom: park_data.zoomLevel
    })
    console.log(park_data.clickedParkCoordinates)
    document.getElementById("parks_information_overlay_container").style.display = "block";
    document.getElementById("park_info_overlay_park_name").innerHTML = park_data.clickedParkName;
    document.getElementById("park_info_overlay_coordinates").innerHTML =
    `${park_data.clickedParkLatitude}, ${park_data.clickedParkLongitude}`;
    showHideOtherNames(park_data.clickedParkOtherNames);
    document.getElementById("park_info_overlay_year_opened").innerHTML = park_data.clickedParkOpeningPeriod;
    document.getElementById("park_info_overlay_year_opened_box").style =
    `background-color: ${park_data.openingPeriodColor}`;
    document.getElementById("park_info_overlay_status").innerHTML = park_data.clickedParkStatus;
    document.getElementById("park_info_overlay_status_box").style =
    `background-color: ${park_data.statusColor}`;
    document.getElementById("park_info_overlay_alterations").innerHTML = park_data.clickedParkAlterations;
    document.getElementById("park_info_overlay_alterations_box").style =
    `background-color: ${park_data.alterationsColor}`;
    historyDescription(park_data.clickedPark, park_data.clickedParkName, park_data.clickedParkHistory);
    mapContainer.addEventListener("click", closeOverlay);
}
```

2. Passing/ making objects over having long lists of variables. By creating an object containing variables (defined elsewhere), it makes the code more readable and easier to reuse.

```
let park_data = {
       clickedParkCoordinates: clickedParkCoordinates,
       zoomLevel: zoomLevel,
       clickedParkName: clickedParkName,
       clickedParkAlterations: clickedParkAlterations,
       clickedParkLatitude: clickedParkLatitude,
       clickedParkLongitude: clickedParkLongitude,
       clickedParkOtherNames: clickedParkOtherNames,
       openingPeriodColor: openingPeriodColor,
       clickedParkOpeningPeriod: clickedParkOpeningPeriod,
       clickedParkStatus: clickedParkStatus,
       alterationsColor: alterationsColor,
       clickedParkHistory: clickedParkHistory
   }
```

02.04.24

Today has been an emotional rollercoaster when it comes to the build but I think it has ended on a relative high and I actually think I have achieved quite a lot. Today I have:

- Created a function that opens the park overlay when a park name is clicked on in the list view. (Part of the functionality was completed on the 29th March, where I got as far as when the park name in the list view is clicked on the map page is opened).

  The first step involved converting the list of variables I created for each of the clicked park properties into a function. This way the 'clicked feature' can be used by both the overlay opened from the map view or the overlay opened from the list view. The information is then stored in an object called 'park_data' which can easily be called upon in both instances. Consultant Kane suggested that I do it this way.

```
function getParkData(clickedFeature){
    let clickedParkName = clickedFeature.properties.name;
    let clickedParkLongitude = clickedFeature.properties.longitude;
    let clickedParkLatitude = clickedFeature.properties.latitude;
    let clickedParkOpeningPeriod = clickedFeature.properties.period_opened;
    let clickedParkStatus = clickedFeature.properties.status;
    let clickedParkAlterations = clickedFeature.properties.alteration;
    let clickedParkOtherNames = clickedFeature.properties.other_names;
    let clickedParkHistory = clickedFeature.properties.brief_history;
    let clickedParkSize = clickedFeature.properties.size;
    let zoomLevel = determineParkZoom(clickedParkSize);
    let longitudeDiscrepancy = determineLongitudeDiscrepancy(clickedParkSize);
    let latitudeDiscrepancy = determineLatitudeDiscrepancy(clickedParkSize);
    let clickedParkCoordinates =
    {lng: clickedFeature.properties.longitude + longitudeDiscrepancy,
     lat: clickedFeature.properties.latitude + latitudeDiscrepancy};
    console.log(clickedParkSize);
    console.log(latitudeDiscrepancy);

    let openingPeriodColor = setOpeningDateColor(clickedParkOpeningPeriod);
    let statusColor = setStatusColor(clickedParkStatus);
    let alterationsColor = setAlterationColor(clickedParkAlterations);

    let park_data = {
        clickedParkCoordinates: clickedParkCoordinates,
        zoomLevel: zoomLevel,
        clickedParkName: clickedParkName,
        clickedParkLatitude: clickedParkLatitude,
        clickedParkLongitude: clickedParkLongitude,
        clickedParkOtherNames: clickedParkOtherNames,
        clickedParkHistory: clickedParkHistory,
        clickedParkStatus: clickedParkStatus,
        statusColor: statusColor,
        clickedParkAlterations: clickedParkAlterations,
        alterationsColor: alterationsColor,
        openingPeriodColor: openingPeriodColor,
        clickedParkOpeningPeriod: clickedParkOpeningPeriod
    }

function checkStorage(){
    var localStorageContent = window.localStorage.getItem("list_view_clicked_park");
    if(localStorageContent){
        var listViewParkData = JSON.parse(localStorageContent);
        var park_data = getParkData(listViewParkData);
        openParkOverlay(undefined, park_data);
        window.localStorage.setItem("list_view_clicked_park", "");
    }
}
```

One thing that is bothering me a little is the load speed of the map when the overlay is loaded from the list page. I spoke to Kane about this and apparently it can't be helped because the way I have structured the website is to have the list view on a seperate page. This means that whenever a park name is clicked on in the list view and the map page is loaded the map will have to load from scratch every time. If I wanted to prevent this I could choose to build the list view on the same page as the map and just have it as a div that is displayed or hidden rather than loading an entirely new page. If I have the time at the end I may restructure the website but only if I have time at the end.

- Updated the function associated with the close button on the overlay so that if the overlay was opened from the map view it closes the overlay and remains on the map page but if the overlay was opened via the list page then it takes the user back to the list page. To achieve this I used the following code:

  The line of code below is added to the javaScript page made for the non-map pages.

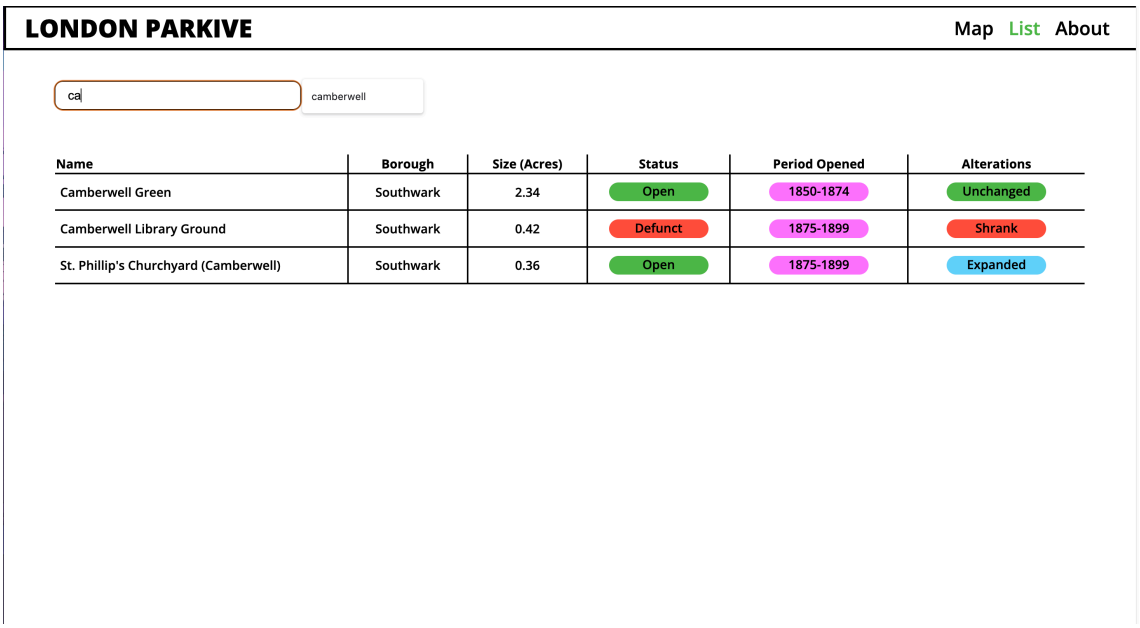
```
window.localStorage.setItem("map_access_mode", "from_list_view");
```

The code below is added to the global scope of the main javascript page.

```
function closeOverlay(){
    let accessMode = window.localStorage.getItem("map_access_mode");
    if(accessMode === "from_list_view"){
        var baseurl = window.location.pathname;
        var spliturl = baseurl.split("london_parkive.html");
        var targeturl = spliturl[0] + "lp_list_page.html";
        console.log(targeturl);
        var targetHref = window.location.origin + targeturl;
        window.open(targetHref, "_self");
        window.localStorage.setItem("map_access_mode", "");
    } else if(parkInfoOverlay.style.display === "block"){
            parkInfoOverlay.style.display = "none";
    };
}
```

- Made the search bar on the list view functional so if you start typing in a park name it filters out all the irrelevant names. The code I used to achieve that can be found below:

```
function searchBarFilter(){
    var input, filter, table, tr, td, i, txtValue;
    input = document.getElementById("list_view_search_bar");
    filter = input.value.toUpperCase();
    table = document.getElementById("london_parkive_list_view");
    tr = table.getElementsByTagName("tr");
    for(i = 0; i < tr.length; i++){
        td = tr[i].getElementsByTagName("td")[0];
        if (td){
            txtValue = td.textContent || td.innerText;
            if (txtValue.toUpperCase().indexOf(filter) > -1){
                tr[i].style.display = "";
            } else {
                tr[i].style.display = "none";
            }
        }
    }
}
```



This screenshot depicts the London Parkive list view page with a functioning search bar that filters out any irrelevant parks.

- The final thing that I achieved today was to remove duplicates from the list view. There are duplicates in the list because some of the larger parks (Peckham Rye Park and Burgess Park) are actually made up of multiple polygons on the map. This meant that duplicates of these parks exist. To remove these duplicates I used the following function:

```
list_view_deduplicated_park_data = parksShapes.features.reduce((accumulator, current) =>{
            if(!accumulator.find((item) => item.properties.name === current.properties.name)){
                accumulator.push(current);
            }
            return accumulator;
        }, []);
```

This was my first time using the **.reduce()** function and I am quite please with myself because I discovered this method on my own (without the help of consultant Kane) by searching Google and I was able to get it working more or less on my own.

🛠️😓.04.24

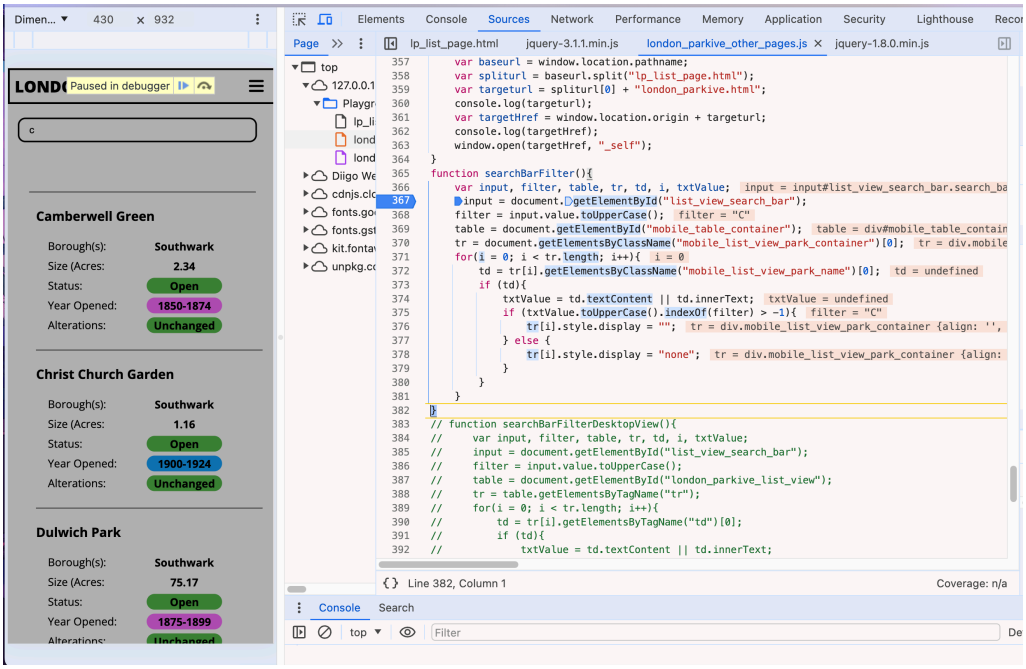
I feel like I've only done a little today, though some of the bits took a lot of working out! Today I have:

- Got the search bar on the list page working for the mobile view as well as the table view. To achieve this I updated the searchBarFilter function I created yesterday. I created an if else statement to determine screen size and depending on the screen width and which list view is present will target the specific elements for each one.

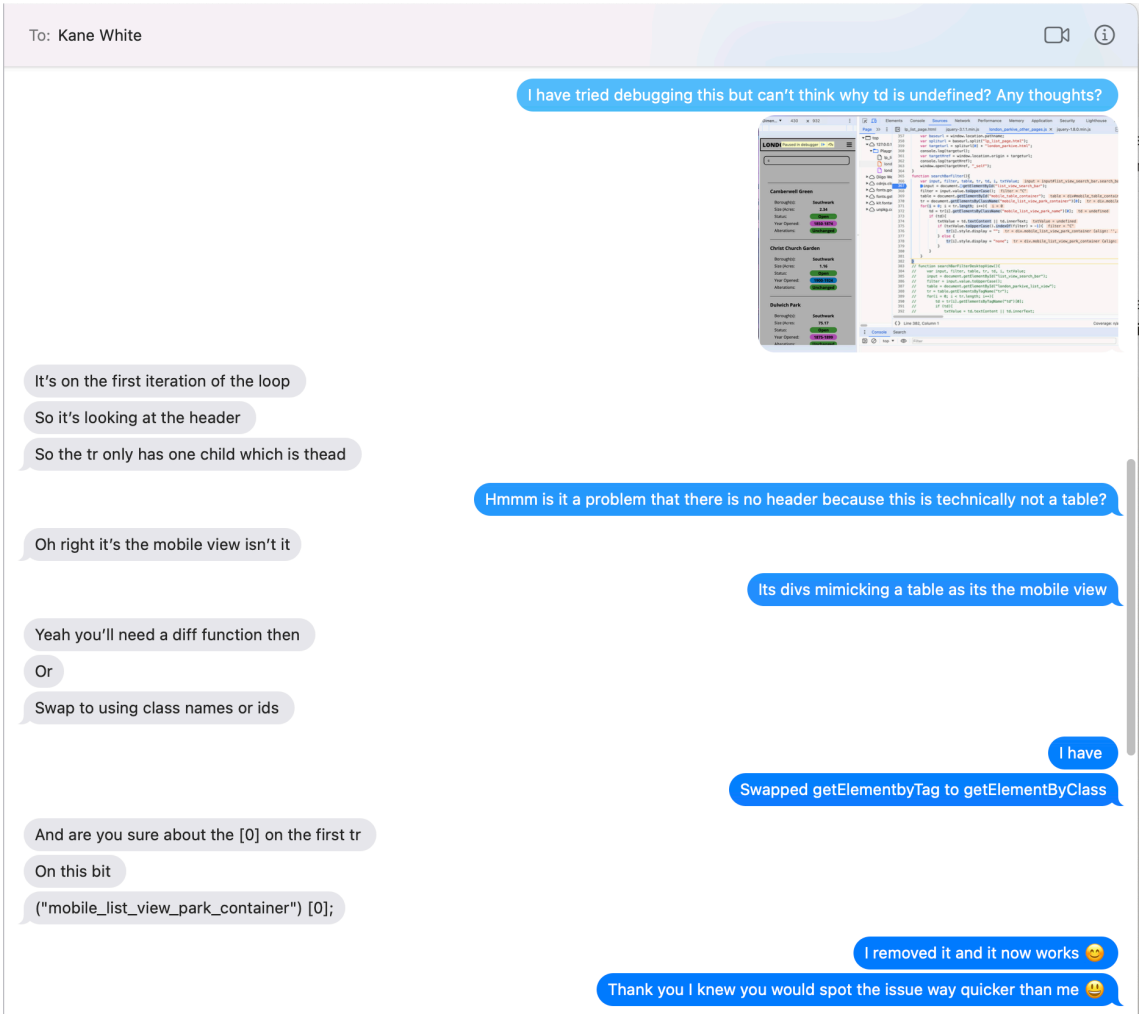
```
function searchBarFilter(){
    var input, filter, table, tr, td, i, txtValue;
    if(screen.width < 900){
        input = document.getElementById("list_view_search_bar");
        filter = input.value.toUpperCase();
        table = document.getElementById("mobile_table_container");
        tr = document.getElementsByClassName("mobile_list_view_park_container");
        for(i = 0; i < tr.length; i++){
            td = tr[i].getElementsByClassName("mobile_list_view_park_name")[0];
            if (td){
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1){
                    tr[i].style.display = "";
                } else {
                    tr[i].style.display = "none";
                }
            }
        }
    }
    else{
        input = document.getElementById("list_view_search_bar");
        filter = input.value.toUpperCase();
        table = document.getElementById("london_parkive_list_view");
        tr = table.getElementsByTagName("tr");
        for(i = 0; i < tr.length; i++){
            td = tr[i].getElementsByTagName("td")[0];
            if (td){
                txtValue = td.textContent || td.innerText;
                if (txtValue.toUpperCase().indexOf(filter) > -1){
                    tr[i].style.display = "";
                } else {
                    tr[i].style.display = "none";
                }
            }
        }
    }

}
```

Part way through I got a little stuck but I did a pretty good job of debugging it before turning to consultant Kane for working out the very last piece of the puzzle. I have always found using the inspect window to debug rather tricky but I am slowly starting to get the hang of it.



Screenshot of the inspect window depicting the debugging process. The bits highlighted in red seem to hold the clues as to what is going wrong.



Screenshot of messages between me and Kane as he helped me debug the problem.

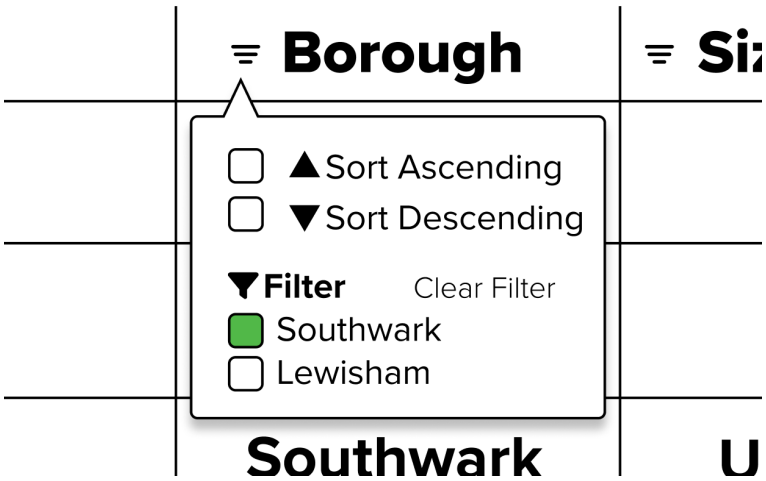
- Created a filter icon using CSS which is dynamically generated so that each table header cell has one. I am getting pretty good at both creating simple CSS icons and generating html elements in javaScript!

Name	Borough	Size (Acres)	Status	Period Opened	Alterations
Camberwell Green	Southwark	2.34	Open	1850-1874	Unchanged
Christ Church Garden	Southwark	1.16	Open	1900-1924	Unchanged

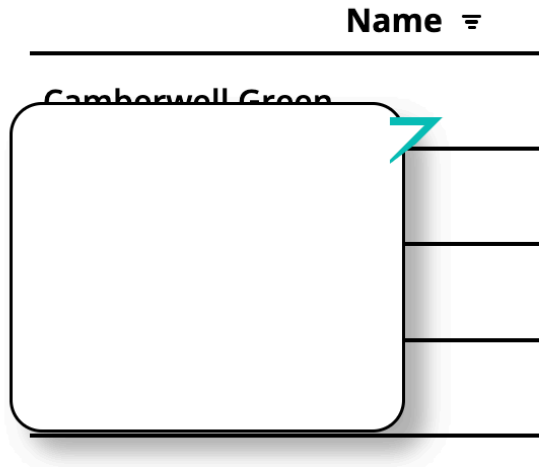
To achieve this I had to create multiple container divs within container divs so I could get the flex styling just right.

```
var th = document.createElement("th");
            headerRow.appendChild(th);
            var thContainerContainer = document.createElement("div");
            thContainerContainer.classList.add("list_view_table_thContainerContainer");
            th.appendChild(thContainerContainer);
            var thContainer = document.createElement("div");
            thContainer.classList.add("list_view_table_thContainer");
            thContainerContainer.appendChild(thContainer);
            thText = document.createElement("div");
            thText.textContent = headerMapping[key];
            thText.classList.add("list_view_table_header_cell_text");
            thContainer.appendChild(thText);
            var filterIconContainer = document.createElement("div");
            filterIconContainer.classList.add("filterIconContainer");
            thContainer.appendChild(filterIconContainer);
            var filterIcon = document.createElement("div");
            filterIcon.classList.add("filterIcon");
            filterIconContainer.appendChild(filterIcon);
            var filterIconLine1 = document.createElement("div");
            filterIconLine1.classList.add("filterIconLine1");
            filterIcon.appendChild(filterIconLine1);
            var filterIconLine2 = document.createElement("div");
            filterIconLine2.classList.add("filterIconLine2");
            filterIcon.appendChild(filterIconLine2);
            var filterIconLine3 = document.createElement("div");
            filterIconLine3.classList.add("filterIconLine3");
            filterIcon.appendChild(filterIconLine3);
            var filterOptionsContainer = document.createElement("div");
            filterOptionsContainer.classList.add("list_view_table_filter_options_container");
            filterIconContainer.appendChild(filterOptionsContainer);
```

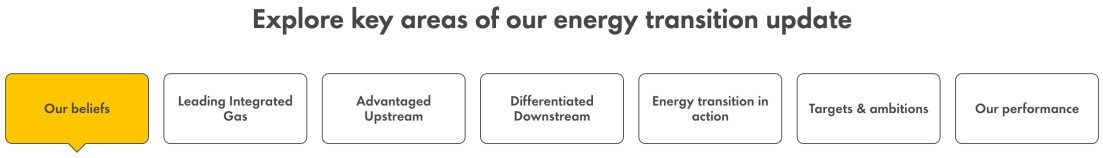
- Next I created a speech bubble shaped div which will contain the filter functionality. Below you can see the Figma designs I have created for this.



This was a real pain to make because of the pointy bit at the top. On the following page you can see one of the failed attempts. The reason why it was so difficult to do is because I wanted a border around the whole shape. Creating the pointy bit itself was pretty straight forward but creating a border around it a total nightmare as it required creating two shapes and trying to line them up!



The method that worked in the end I found on Shell's website. As I sat on the couch procrastinating for 10 minutes I was scrolling through LinkedIn when I came across an article on Shell's website. As I was reading it I noticed that on their website they have these speech bubble style boxes with a border around them! So I went to the inspect panel and had a little look and copied and pasted the CSS for those boxes and customised it to fit my own website. The result was a total success!



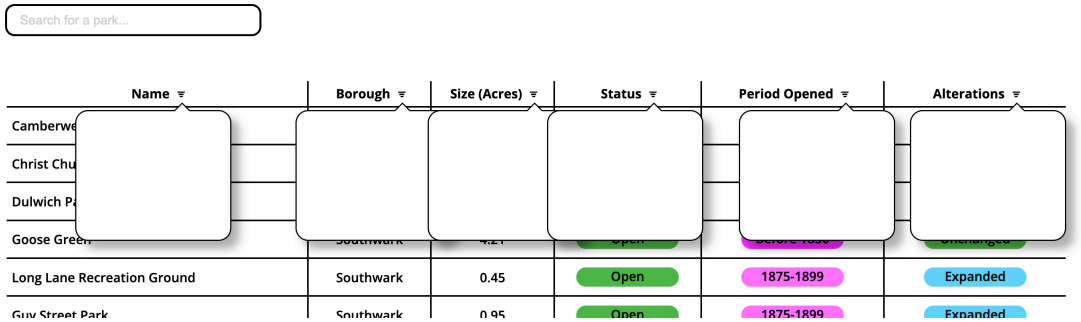
Screenshot of the speech bubbles I found on Shell's website.

The CSS for the filter options container can be found below:

```
.list_view_table_filter_options_container{
    width: 12rem;
    height: 10rem;
    border-radius: 1rem;
    border-style: solid;

    background-color: white;
    position: absolute;
    right: -3rem;
    top: 2.2rem;
    box-shadow: 10px 10px 10px rgba(0,0,0,0.3);
}
.list_view_table_filter_options_container:before{
    content: "";
    position: absolute;
    top: -1px;
    left: 50%;
    transform: translate(130%, -100%) rotate(180deg);
    display: block;
    width: 0;
    height: 0;
    border: 10px solid transparent;
    border-top-color: black;
}
.list_view_table_filter_options_container:after{
    content: "";
    position: absolute;
    bottom: -1px;
    left: 50%;
    transform: translate(130%, -800%) rotate(180deg);
    display: block;
    width: 0;
    height: 0;
    border: 10px solid transparent;
    border-top-color: white;
}
```

The next thing I worked on was positioning the filter containers. Initially I had just used position absolute but what I soon realised was that when the window was resized it was no longer lining up with the filter icon. So I did some research into different positions and found out that if a parent element also has a non-static position then the absolute of the child element will be in relation to the parent element rather than the page as a whole. So I dynamically generated yet another container (the reason for this was that if I used the filter icon div as a parent element the hover effect applied to each line in the icon, which is targeted by the fact they are divs would also affect the filter options container, meaning if you hovered over it it would turn green) and set the position to relative. Now the position absolute on the options container is in relation to the icon rather than the whole page. And that's as far as I got today!



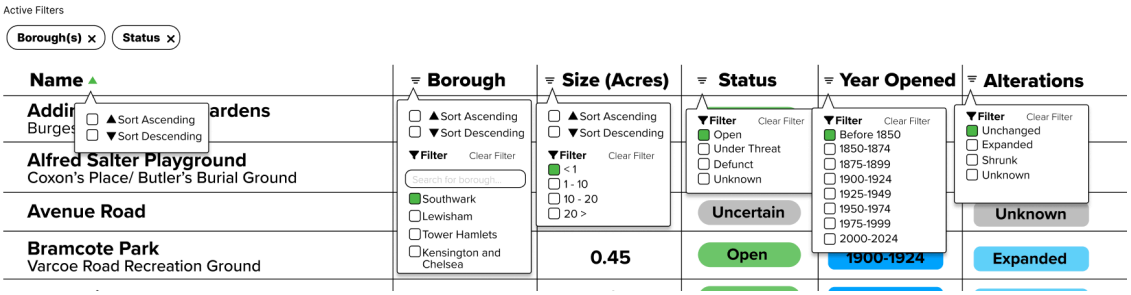
🛠️🎨04.04.24

Creating the filter functionality continues. This has been such a long process and I feel like I am achieving very little everyday. But I am learning a ton of useful stuff so that is what I need to focus on! Today I have:

- Assigned classnames and ids to each of the filter containers that I created yesterday. I achieved this by using a Switch function.

```
switch(thText.textContent){
            case "Name":    filterOptionsContainer.classList.add("filter_type_sort");
                            filterOptionsContainer.id = "name_Filter_Container";
            break;
            case "Borough":     filterOptionsContainer.classList.add("filter_type_sort_filter_with_search");
                                filterOptionsContainer.id = "borough_Filter_Container";
            break;
            case "Size (Acres)" :   filterOptionsContainer.classList.add("filter_type_sort_filter");
                                    filterOptionsContainer.id = "size_Filter_Container";
            break;
            case "Status":  filterOptionsContainer.classList.add("filter_type_filter");
                            filterOptionsContainer.id = "status_Filter_Container";
            break;
            case "Period Opened" :    filterOptionsContainer.classList.add("filter_type_filter");
                                    filterOptionsContainer.id = "period_opened_Filter_Container";
            break;
            case "Alterations" :    filterOptionsContainer.classList.add("filter_type_filter");
                                    filterOptionsContainer.id = "alterations_Filter_Container";
            break;
            default:
            break;
        }
```

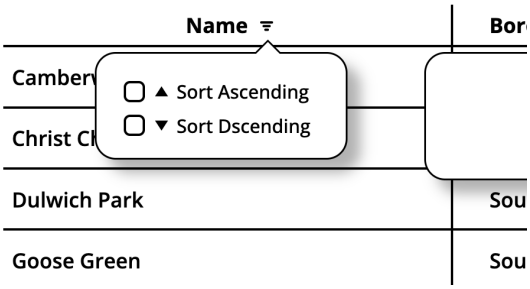
- Designed what each filter container should look like in Figma:



- Managed to create the elements of the Name dropdown filter. I have been trying to set up the code so it will be easier to apply the same elements across all of the drop downs. For example the "Sort Ascending" and "Sort Descending" elements can be found on multiple dropdowns so I have created a function to generate these. This should streamline the process and shorten the amount of code required going forward.

```
filterIconContainer.appendChild(filterOptionsContainer);
          var filterOptionsContent = document.createElement("div");
          filterOptionsContent.classList.add("lvt_filter_options_content");
          filterOptionsContainer.appendChild(filterOptionsContent);
          let lvt_filter_content_row = document.createElement("div");
          lvt_filter_content_row.classList.add("lvt_filter_content_row");
          let lvt_filter_checkbox = document.createElement("div");
          lvt_filter_checkbox.classList.add("lvt_filter_checkbox");
          let lvt_filter_ascending_arrow = document.createElement("div");
          lvt_filter_ascending_arrow.classList.add("lvt_filter_ascending_arrow");
          let lvt_filter_descending_arrow = document.createElement("div");
          lvt_filter_descending_arrow.classList.add("lvt_filter_descending_arrow");
          let lvt_filter_key_ascending = document.createElement("div");
          lvt_filter_key_ascending.classList.add("lvt_filter_key");
          lvt_filter_key_ascending.innerHTML = "Sort Ascending";
          let lvt_filter_key_descending = document.createElement("div");
          lvt_filter_key_descending.classList.add("lvt_filter_key");
          lvt_filter_key_descending.innerHTML = "Sort Dscending";
          function sortGenerator(){
              lvt_filter_content_row.appendChild(lvt_filter_checkbox);
              lvt_filter_content_row.appendChild(lvt_filter_ascending_arrow);
              lvt_filter_content_row.appendChild(lvt_filter_key_ascending);
              filterOptionsContent.appendChild(lvt_filter_content_row);
              lvt_filter_content_row = document.createElement("div");
              lvt_filter_content_row.classList.add("lvt_filter_content_row");
              lvt_filter_checkbox = document.createElement("div");
              lvt_filter_checkbox.classList.add("lvt_filter_checkbox")
              lvt_filter_content_row.appendChild(lvt_filter_checkbox);
              lvt_filter_content_row.appendChild(lvt_filter_descending_arrow);
              lvt_filter_content_row.appendChild(lvt_filter_key_descending);
              filterOptionsContent.appendChild(lvt_filter_content_row);
          }
          if(filterOptionsContainer.classList.contains("filter_type_sort")){
              sortGenerator();
          }
```

Something I have learned today is a distinction between using **`var`** and **`let`**. By using **`let`** in this instance I have been able to reuse and redefine variables in the code. Whereas when I used **`var`** I was unable to do this.

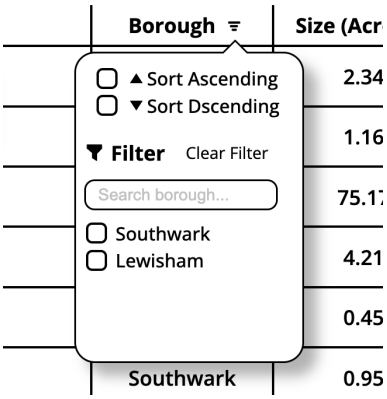


I feel like I am climbing a mountain when it comes to the amount of stuff that is left to do on the list page! In my initial planning I anticipated it would take a week to create this page. In reality I have already spent 2 weeks on it and I am nowhere near finished with it. I just keep reminding myself that this is probably one of the hardest sections of the whole website to code and I am learning a ton of useful stuff so it is worth spending the time on this because everything else after this section should (in theory) be crazy easy!

🛠️05.04.24

Again, I feel like I have only done a small amount of work today but it is crucial work which will speed up the process going forwards. Today I have:

- Created the visuals for the Borough Filter Dropdown.



This took a bit of working out as rather than adding each borough statically I decided to create a dynamic way of adding them so that in the future if more boroughs were added it would automatically update the number of boroughs. This was achieved by using the **.reduce()** function which I learned pretty recently to filter through the boroughs on the JSON file and create a new array of objects that don't include any duplicates. In this instance as it was filtering through the borough section and there are only two boroughs currently on the JSON list only two objects are in the array.

```
var lvt_filter_borough_keys;

            lvt_filter_borough_keys = parksShapes.features.reduce((accumulator, current) =>{
            if(!accumulator.find((item) => item.properties.borough === current.properties.borough)){
                accumulator.push(current);
            }
            return accumulator;
        }, []);
```

Then the next step is to create a **.forEach()** function which will loop through each instance of this new array and create a series of elements accordingly:

```
let lvt_filter_key_name;
            lvt_filter_borough_keys.forEach(feature =>{
                lvt_filter_content_row = document.createElement("div");
                lvt_filter_content_row.classList.add("lvt_filter_content_row");
                lvt_filter_checkbox = document.createElement("div");
                lvt_filter_checkbox.classList.add("lvt_filter_checkbox");
                lvt_filter_content_row.appendChild(lvt_filter_checkbox);
                lvt_filter_key_name = document.createElement("div");
                lvt_filter_key_name.classList.add("lvt_filter_key");
                lvt_filter_key_name.innerHTML = feature.properties.borough;
                lvt_filter_content_row.appendChild(lvt_filter_key_name);
                lvt_filter_filter_section.appendChild(lvt_filter_content_row);
            });
```

Tomorrow (or Monday if I decide to take the weekend off) I would like to streamline this into a function that can be reused for all of the other filter dropdowns with filters.

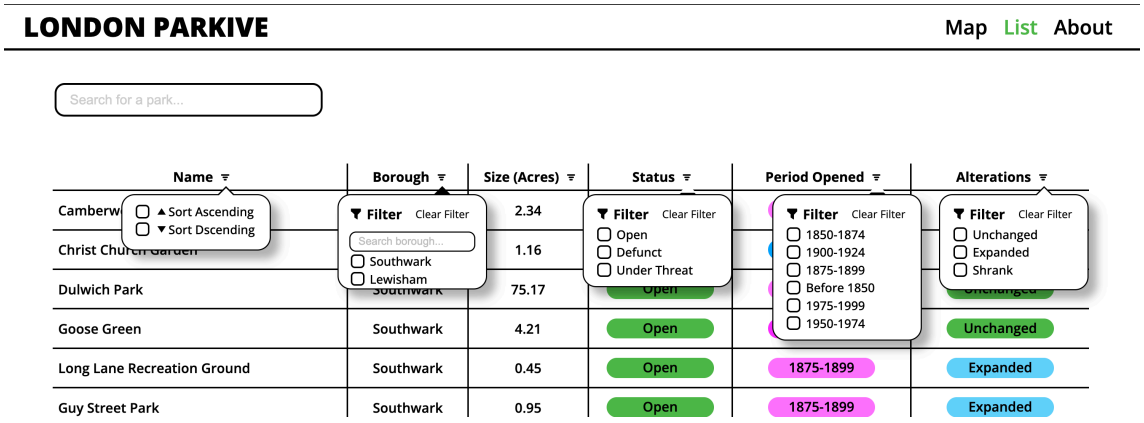
- The second thing I did today was to streamline the code that I wrote today and yesterday. I realised that a lot of the dynamically generated content was very repetitive so I decided to create a function for element that is being dynamically generated so that I am not creating dupluicates of code over and over again. I haven't finished doing this yet so will spend tomorrow (or Monday) continuing with this. While it is a tedious process I think it will save me time in the future and it has also made my code much more legible and easier to manage.

Today was another slow day but I am yet another step closer to the completion of the list page.

🛠️08.04.24

Today was another slow and tricky day. I have realised how poorly organised my code is but fear that reorganising it may cause more issues. But Kane has offered to do a code review with me so he can talk through how I could have, retrospectively, better organized my code for next time. With all of this in mind this is what I got up to today:

- Managed to generate the content for each of the filter containers (apart from the size container). The section of code that executes this can be found on the following page though you can't see the referenced functions (I think I have previously posted some of the code snippets for them):



```
let borough = "borough";
        let status = "status";
        let period_opened = "period_opened";
        let alteration = "alteration";
        switch(filterOptionsContainer.id){
            case "name_Filter_Container":
               let nameDropdownContent = sortFilterGenerator();
            break;
            case "borough_Filter_Container":
                let filterWithSearchSection = createFilterwithSortFilterAndSearch();
                let boroughData = createFilterKeysData(borough);
                let boroughKeySection = createFilterKeysSection(boroughData, borough);
                console.log(boroughData);
            break;
            case "size_Filter_Container":
            break;
            case "status_Filter_Container":
              let statusDropdownContent = filterFilterGenerator(status);
            break;
            case "period_opened_Filter_Container":
                let periodOpenedDropdownContent = filterFilterGenerator(period_opened);
            break;
            case "alterations_Filter_Container":
                let alterationDropdownContent = filterFilterGenerator(alteration);
            break;
            default:
            break;
        }
```

- After completing this I started working on hiding and displaying this code. This is where I began to realise how messy my code is. Half way through the day I started to try and reorganise the code but this caused many errors so I put the code back to how it was and proceeded. The code below is the code that worked:

```
lvt_filter_dropdown_container.addEventListener("click", function(e){
            console.log(e);
            console.log(e.target.classList);

            if (e.target.classList.contains("id_name")){
                console.log("Name Filter Clicked");
                let nameContainer = document.getElementById("name_Filter_Container");
                if(nameContainer.style.display == "none"){
                    document.getElementById("name_Filter_Container").style.display = "block";
                }
                else{
                    document.getElementById("name_Filter_Container").style.display = "none";
                }
            }
            else if(e.target.classList.contains("id_borough")){
                console.log("Clicked");
                let nameContainer = document.getElementById("borough_Filter_Container");
                if(nameContainer.style.display == "none"){
                    document.getElementById("borough_Filter_Container").style.display = "block";
                }
                else{
                    document.getElementById("borough_Filter_Container").style.display = "none";
                }
            }
```

The code is actually longer but I have just showed two examples of the if else statement. The rest of the code follows a similar pattern. Initially the function did not contain the parameter of ‘e’ and the if else statements were written as:

```
if (headerFilterIcon.classList.contains("id_name")){
```

This didn’t work. Consultant Kane then told me that it doesn’t work because headerFilterIcon is used on all of the icons and as the function loops through to generate the table “id_name” is being overwritten by the other class names. Long story short by writing:

```
if (e.target.classList.contains("id_name")){
```

Instead it targets the icon that was clicked specifically thus making it work. A minor issue that I might try to work on tomorrow is that the icon when clicked is not very responsive. Sometimes you have to click it multiple times before it works. From a UX perspective this is very annoying and is the kind of thing that would irritate me as a user. So I would like to work on fixing this. Another thing I want to work on tomorrow is reorganizing my code a little. Specifically I want to make the table headers statically in HTML rather than dynamically generating it.

🛠️09.04.24

Today has been a rollercoaster of a day. Full of ups and downs! But it has also been a very productive day. The development of the list view page continues. Today I:

- Made the header row for the table static and in doing so cut down the javaScript code by a few hundred lines. A big lesson I have learned is :

> 💡 **generating stuff using javaScript isn't always the most efficient way to do things!**

I have spent ages (probably the better part of a week or so) trying to generate as much content as possible using javaScript. Today, in the space of a morning I was able to recreate the same amount of content in a couple of hours using static HTML. Statically creating the header elements, which includes the filter dropdown menus will make it much easier to add functionality to the various elements later down the line.

- Started adding functionality to the filters. Today I managed to add functionality to the Name filter, which sorts the order of the table alphabetically in either ascending or descending order.and the Size filter which arranges the table in ascending or descending order of size. To achieve this I used the following code:

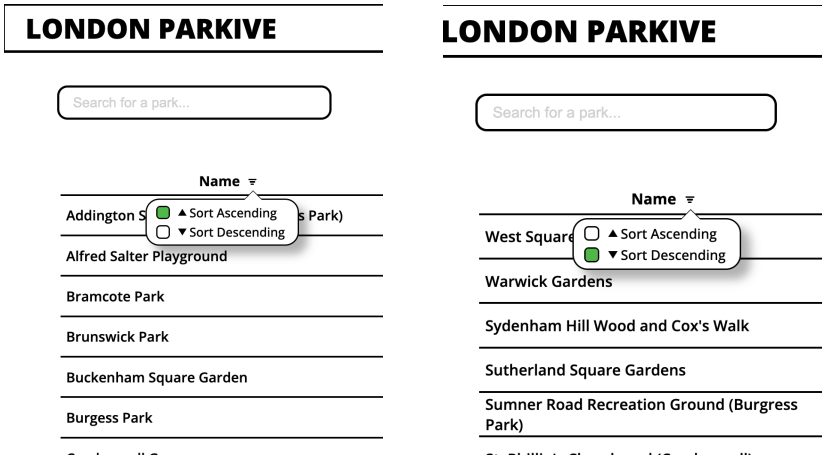
```
function activateSizeAscendingFilter(){
      console.log("clicked");
      if(size_filter_ascending_checkbox_active_state.style.display === "none"){
          size_filter_ascending_checkbox_active_state.style.display = "block";
          document.getElementById("list_view_table_body").remove();
          ascendingSize_tablebody = document.createElement("tbody");
          ascendingSize_tablebody.id = "list_view_table_body";
          list_view_table.appendChild(ascendingSize_tablebody);
          list_view_deduplicated_park_data.sort((a,b) => a.properties.size > b.properties.size ? 1 : -1 );
          ascendingSize_tablebody_content =
          generateTable(list_view_deduplicated_park_data, tableProperties, ascendingSize_tablebody);
          if(size_filter_descending_checkbox_active_state.style.display === "block"){
              size_filter_descending_checkbox_active_state.style.display = "none";
          }
      }
      else{
          size_filter_ascending_checkbox_active_state.style.display = "none";
      }
  }
```

In a nutshell, when the checkbox is clicked, in this instance the “Sort Ascending” checkbox, first of all the inside of the checkbox is coloured green (and if the “Sort Descending “ checkbox is also active this is deactivated). The next step is to remove the table body contents of the table. Without doing this when the generateTable function is called the contents of the table won’t change (It took me a mini-meltdown and a lot of time to work this one out). As the **.remove()** function removes the **<tbody>** element from the DOM the next step is to re-add this element but under a new variable name. In this instance “ascendingSize_tablebody” because this best describes the context that the new table body is being used in. The original class name “list_view_table_body” is reassigned to the new tbody element etc. Once the new table body is in place it is time to generate a new table body but not before reordering the table contents in ascending alphabetical order. I am yet to completely wrap my head around this next function. I discovered this method with the combined help from ChatGPT and Consultant Kane but it works.

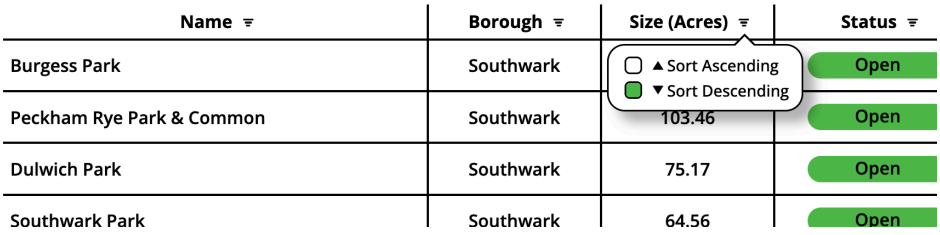
```
list_view_deduplicated_park_data.sort((a,b) => a.properties.size > b.properties.size ? 1 : -1 );
```

By swapping the **1 : -1** to **-1 : 1** the order that it is sorted in is flipped. The same code is applied to the descending size checkbox, the ascending name checkbox and the descending checkbox just with different parameters.

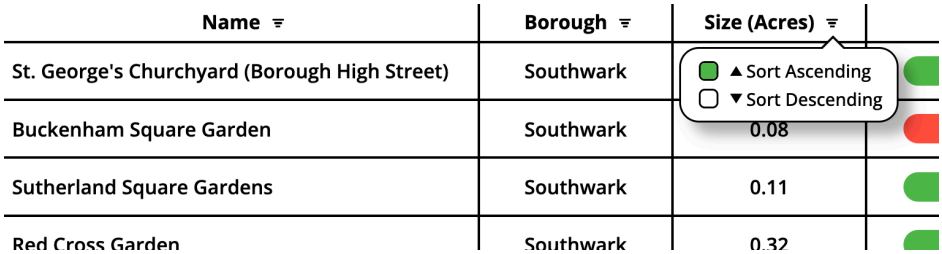
The list view page is still a long way from being finished but I am slowly getting there. I feel like the tortoise in the heir and the tortoise story. Slowly wins the race.



The screenshot on the left depicts the “Sort Ascending” filter in action (note how the listed parks are in ascending order). The screenshot on the right depicts the “Sort Descending” filter in action (note how the listed parks are in descending order).



Screenshot depicting the “Sort Descending” size filter in action.



Screenshot depicting the “Sort Ascending” size filter in action.

🛠10.04.24

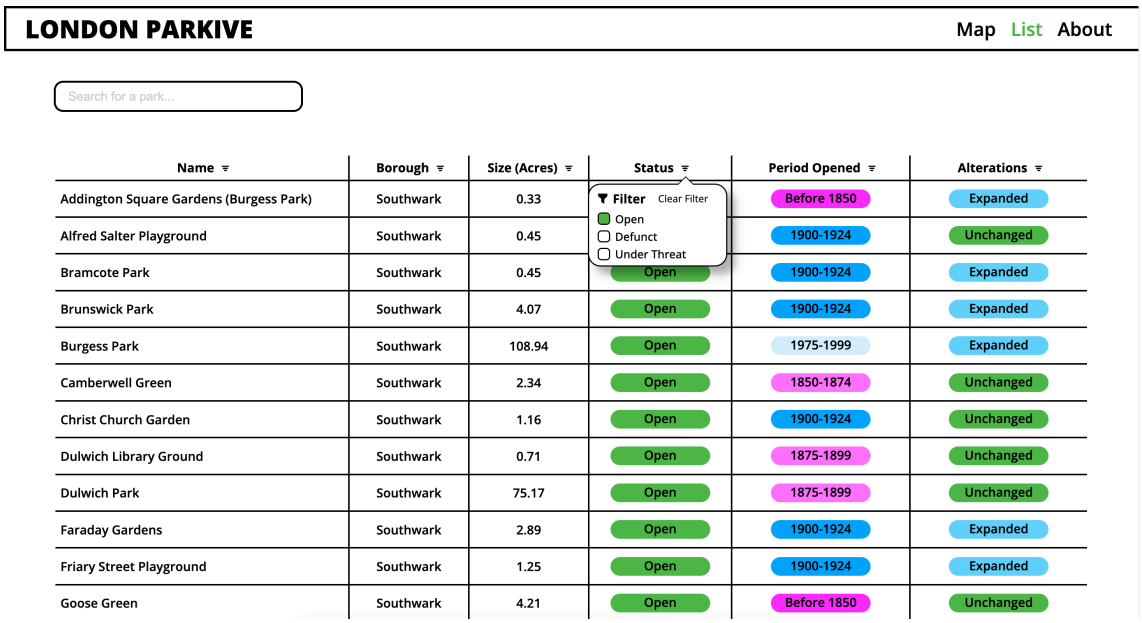
Another slow but productive day. I am getting a little worried about whether I will hit the deadline... Anyway, by the end of today I was able to create the Status Filter with two potential methods to proceed with filtering. The first option I came up with myself with much help from Chat GPT and a little help at the very end from Consultant Kane.

```
let table = document.getElementById("london_parkive_list_view");
            let tr = table.getElementsByTagName("tr");
            let statusBox;
            let txtValue;
            for(i = 1; i < tr.length; i++){
                statusBox = table.getElementsByClassName("status_column")[i];
                if(statusBox){
                    txtValue = statusBox.textContent || statusBox.innerText;
                    if(txtValue === "Open"){
                        tr[i + 1].style.display = "";
                    } else {
                        console.log('setting ', txtValue, tr[i+1].children[0].innerHTML, ' to be hidden');
                        tr[i + 1].style.display = "none";
                    }
                }
            }
```

I loosely based this code off of the search bar function. It took a while to figure out a configuration that would work and the trickiest bit was working out that because the first 'tr' is the table header I have had to start the for loop at 1 rather than zero and because the "status_column" divs position 0 is the 'tr's position 1 having to add 1 ([i+1]) so that they align in the iterations. I had got to a point where statusBox and txtValue were both being recognised in the console as you can see below but because the last iteration was coming back undefined (I think because the iterations for each thing were not lined up) it was causing an error. This was solved by wrapping the display part of the function in an 'if(statusBox){}'. Both of these bits were worked out with the help of Consultant Kane. But I was pleased that I worked out the bulk of the function on my own.

```
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
statusBox:   <div class="list_view_box status_column" style="background-color: rgb(80, 184, 72);">Open</div>
txtValue: Open
```

textValue and StatusBox were being defined in the console but were undefined in the source section.



Screenshot of the working status filter function (yay!) However the functionality to undo the filter has not yet been applied...

In the evening Kane showed me another potential method for creating the filter function. The code for it can be seen below:

```
let open_parks = list_view_deduplicated_park_data.filter((park) =>
park.properties.status == 'Open' && park.properties.year_opened == 'Before 1850');
```

This isn't the complete code but what this code will do is filter through the properties of the JSON file that is used to generate the table and will filter out the data that doesn't fit the parameters, in the above example it will only add the parks with relevant data to the 'open_parks' variable so that when you regenerate the table it will have excluded the irrelevant information. I am yet to determine which method will work best for the overall table.

🛠11.04.24

Today I have continued adding the filter functionality. I have decided to go with the method that Consultant Kane has recommended as I think in the long run it is the more streamlined and scalable way to filter out the items. I have converted this into a reusable function which you can see below:

```
function filterContent(key,keyName){
       let filtered_data = list_view_deduplicated_park_data.filter((feature) => feature.properties[key] == keyName);
       document.getElementById("list_view_table_body").remove();
           newTable = document.createElement("tbody");
           newTable.id = "list_view_table_body";
           list_view_table.appendChild(newTable);
           let newTableContent = generateTable(filtered_data, tableProperties, newTable);
           return newTableContent;
   }
```

This function can now be easily reused for any of the filters. I also created a 'removeFilter()' function which reverses the filter. For the time being I have decided to keep the filtering system simple so you can only have one filter applied at a time across the whole table. I may change this if I have the time at the end of the project to be more versatile.

```
function removeFilter(){
       document.getElementById("list_view_table_body").remove();
       newTable = document.createElement("tbody");
       newTable.id = "list_view_table_body";
       list_view_table.appendChild(newTable);
       newTableContent = generateTable(list_view_deduplicated_park_data, tableProperties, newTable);
       return newTableContent;
   }
```

I have then applied this functionality to the defunct and under threat keys within the status filter dropdown menu and the period opened filter section as well. All in all a very quantitatively productive day!

🛠️12.04.24

Today was a bit of a disastrous day! It started off well as I managed to finish creating most of the filters HOWEVER, I made a start on the borough filter and everything started going wrong. I think in part because I was tired and in part because I was super eager to test out some code that I thought would work but my lack of concentration due to being tired meant that I introduced some bugs, one of which was a missing bracket or curly brace which took me a very long time to find. I even tried asking chatGPT if it could scan through my code to see if it could find it but it couldn't. After hours and many a thought along the lines of "Why the hell am I trying to learn how to code?!" I found the missing element in a fairly obvious spot. My code is still broken (it appears that the html and javascript have some how disconnected but I can't see an obvious reason how? But this is a problem for another time.

🛠️15.04.24

Today has felt like a very productive day! I started the day by writing a to-do list of all the things I want to get done by the Viva. I tried to keep the list as short as possible. Writing the list made me realise just how many features there are that I want to add that I just won't have the time before the viva! But the important thing to remember is that they aren't expecting to see a finished product and therefore it doesn't matter if it isn't finished. Anyhow, the things that I managed to complete off the list today were:

- List Page -> Unknown checkbox doesn't uncheck when another checkbox is ticked.
- List Page -> Name filter Descending checkbox -> should be unselected when the page opens.
- List Page -> Add bottom-margin to table container.
- List Page -> Borough Dropdown -> Make the checkboxes functional and remove search bar. Fix the speech bubble arrow boarder.
- List Page -> Remove "Clear Filter" buttons.
- Map Page -> Search Bar

Most of these were fairly small tasks mainly consisting of fixing minor bugs. I have not finished the search bar yet but I have made a good start on it. I will talk through each of the items briefly.

List Page -> Unknown checkbox doesn't uncheck when another checkbox is ticked

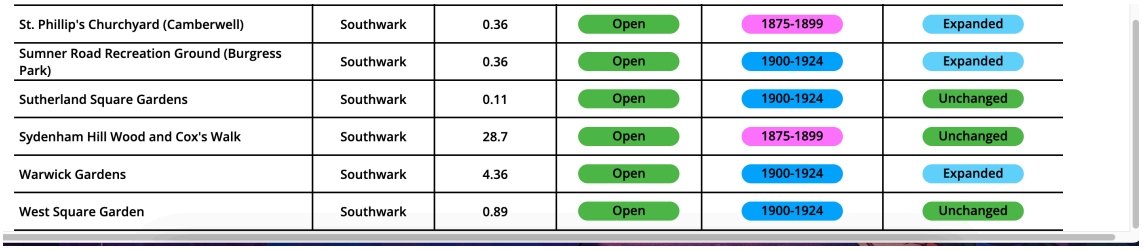
This was a simple solution. The unknown checkbox in question is from the Status Filter Section and the problem was that I had created a new function called **deactivateCheckbox()** which I had made after making the status Filter functions. I had applied it to the other filters but hadn't applied it to the status filter. The status filters meanwhile had a more primitive method for deactivating the checkboxes which involved targetting the specific elements and changing the display to none. As I had added the unknown checkbox later down the line, I had not updated the code to include the unknown checkbox in the list of targeted elements. The problem was solved by adding the **deactivateCheckbox()** function to each of the status filter functions.

List Page -> Name filter Descending checkbox -> should be unselected when the page opens

Another simple fix. For this I just switched the display to "none". Problem solved.

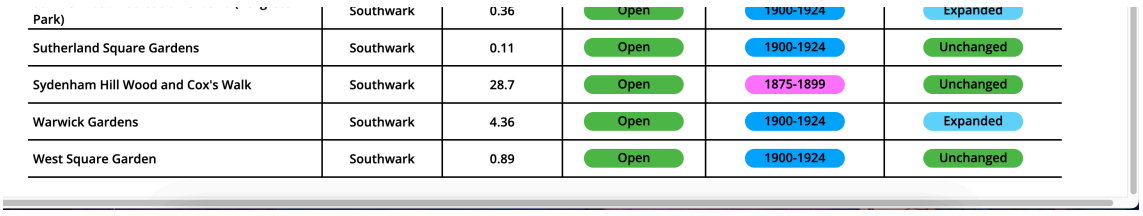
List Page -> Add bottom-margin to table container

This was a little trickier than anticipated. Initially I thought it would be as simple as adding **margin-bottom** on the table element. However this did nothing, I think because the table had **position: absolute;** which for whatever reason meant it wouldn't add a margin to the bottom (I guess cause absolute doesn't follow the rules of the other elements?) So I tried adding the **margin-bottom** to the parent container but this also didn't work. In the end the solution that did work was to add the **position: absolute;** to the parent container (along with the top and left properties) and then add the **margin-bottom** to the table. This worked and I was able to add a bit of margin at the bottom of the table.



St. Phillip's Churchyard (Camberwell)	Southwark	0.36	Open	1875-1899	Expanded
Sumner Road Recreation Ground (Burgress Park)	Southwark	0.36	Open	1900-1924	Expanded
Sutherland Square Gardens	Southwark	0.11	Open	1900-1924	Unchanged
Sydenham Hill Wood and Cox's Walk	Southwark	28.7	Open	1875-1899	Unchanged
Warwick Gardens	Southwark	4.36	Open	1900-1924	Expanded
West Square Garden	Southwark	0.89	Open	1900-1924	Unchanged

Table without margin-bottom

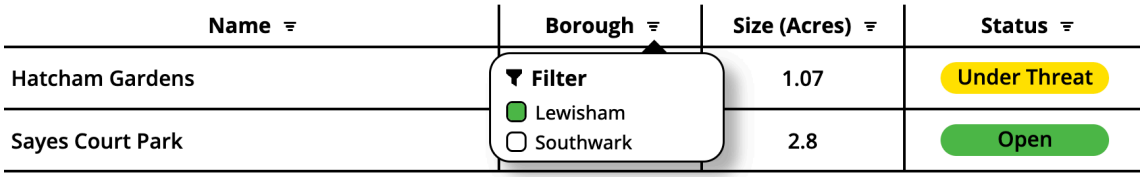


Park)	Southwark	0.36	Open	1900-1924	Expanded
Sutherland Square Gardens	Southwark	0.11	Open	1900-1924	Unchanged
Sydenham Hill Wood and Cox's Walk	Southwark	28.7	Open	1875-1899	Unchanged
Warwick Gardens	Southwark	4.36	Open	1900-1924	Expanded
West Square Garden	Southwark	0.89	Open	1900-1924	Unchanged

Table with margin-bottom

List Page ->Borough Dropdown -> Make the checkboxes functional and remove search bar. Fix the speech bubble arrow boarder

I have decided that given the time constraint I would simplify the borough filter. Originally it had a search bar and I was going to dynamically generate the content but I have decided not to do either given the time constraints. There are more important things to do! I have removed the search bar and statically generated the checkboxes and keys in the same way I have done for all of the other filters. This has saved me hours of time!



This screenshot shows the simplified filter menu and the "Lewisham" filter at play.

List Page -> Remove "Clear Filter" buttons

Another simplification was to remove the "Clear Filter" buttons that were on each filter dropdown menu. Again, I have run out of time to make these. I thought about creating one "Clear Filters" button which would live between the search bar and table but I soon realised that this was a little too complicated given the time constraints.

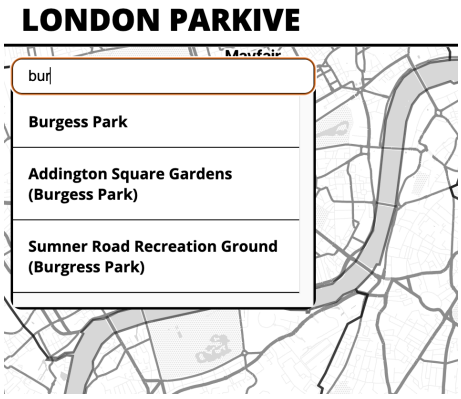
Map Page -> Search Bar

This has been the biggest achievement of today! With the help of chatGPT I was able to make a start on creating the functionality for the search bar on the map page! I have gotten as far as the names of relevant parks popping up underneath the input field when you type letters in to the input field. I should be able to finish this off tomorrow by making it so that when a name is clicked in the search results it will take you to the overlay page. The code for this functionality can be seen below:

```
const searchInput = document.getElementById("map_view_search_bar");
const searchResults = document.getElementById("search_bar_results");


function filterParks(searchTerm){
    return parkNames.filter(feature => feature.toUpperCase().includes(searchTerm.toUpperCase()));
}
function displaySearchResults(results){
    const resultList = results.map(result => `<div class="searchResultsRow">${result}</div>`).join('');
    searchResults.innerHTML = `<div class="searchResults">${resultList}</div>`;
    searchResults.style.display = results.length ? "block" : "none";
}
searchInput.addEventListener("input", function(){
       const searchTerm = this.value.trim();
       console.log(searchTerm);
       if(searchTerm !== ''){
        const filteredParks = filterParks(searchTerm);
        displaySearchResults(filteredParks);
       } else{
        searchResults.style.display = "none";
       }
});
document.addEventListener("click", function(event){
    if(!searchInput.contains(event.target) && !searchResults.contains(event.target)){
        displaySearchResults.style.display = "none";
    }
});
searchResults.addEventListener("click", function(event){
    if(event.target.className === "searchResultsRow"){
        let searchParkName = event.target.textContent;
        searchInput.value = searchParkName;
        searchResults.style.display = "none";


    }
})
```



Screenshot of working search bar

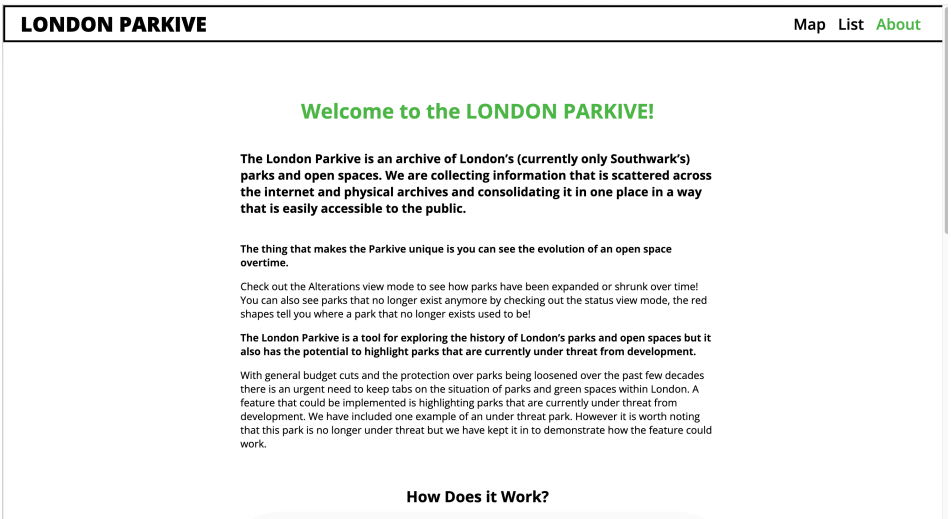
🛠16.04.24

I have completed the to do list (yay!) Today I finished the search bar on the map page. This is now fully functional! I have decided not to create the mobile view filter functionality as it will potentially take a long time to build and could introduce new bugs. I have decided that for the time being I will stop programming and focus on preparing for the viva. If I have the time I will continue designing some other potential features on Figma but I think I will leave the website as it is.

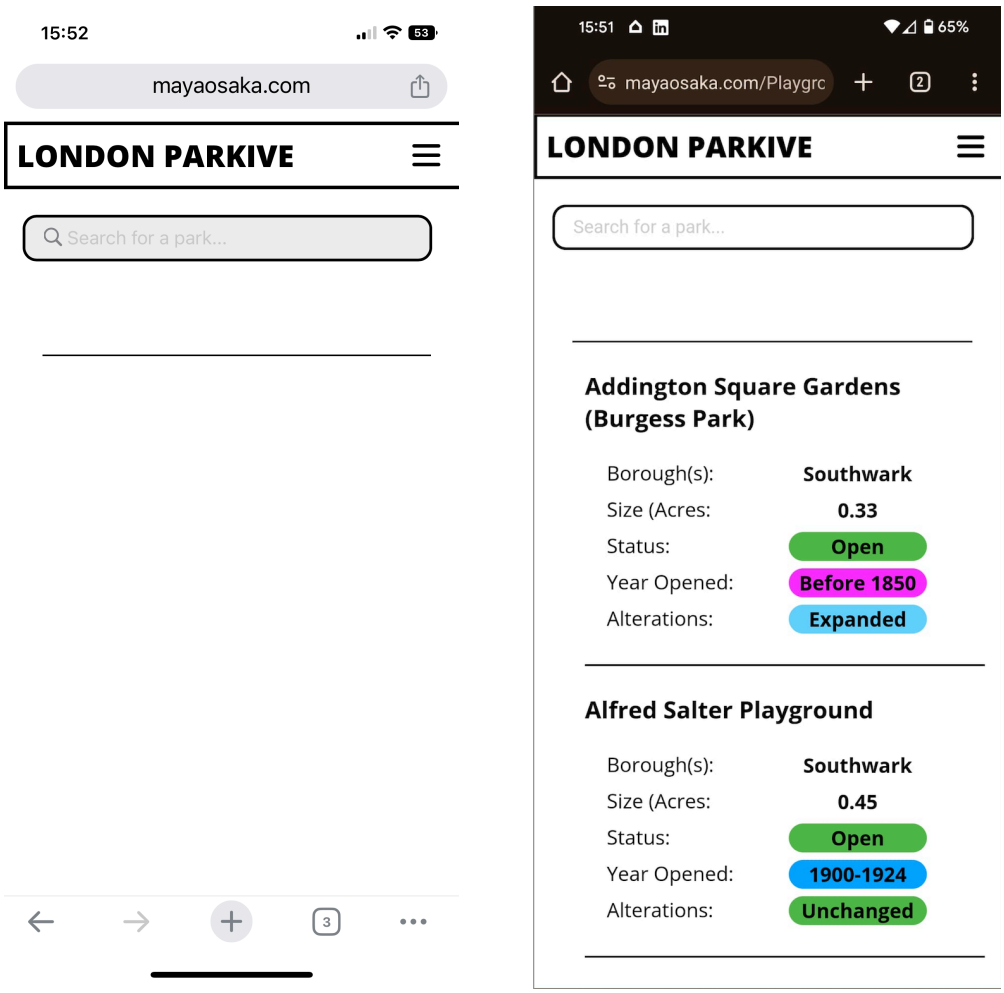
The code for the functioning search bar can be found below:

```
searchResults.addEventListener("click", function(event){
    if(event.target.className === "searchResultsRow"){
        let searchParkName = event.target.textContent;
        searchInput.value = searchParkName;
        searchResults.style.display = "none";
        let targetPark = deduplicatedParks.filter((park) => park.properties.name == searchParkName);
        console.log(targetPark[0]);
       let data = getParkData(targetPark[0]);
       console.log(data);
       openParkOverlay(undefined, data);
    }
```

This is the code that I wrote yesterday but with additional functionality added in. I also created the about page. The code for that is static and uninteresting so I won't bother writing about it but you can see a screenshot of the page below:



One potential bug that I need to look into is that for some reason on my phone and iPad specifically the list view page doesn't work. However I have asked other people to test it on their phones and it works so could this just be an anomaly?



The screenshot on the left shows what the list page view looks like on my phone while the screenshot on the right shows what it looks like on my cousins phone.