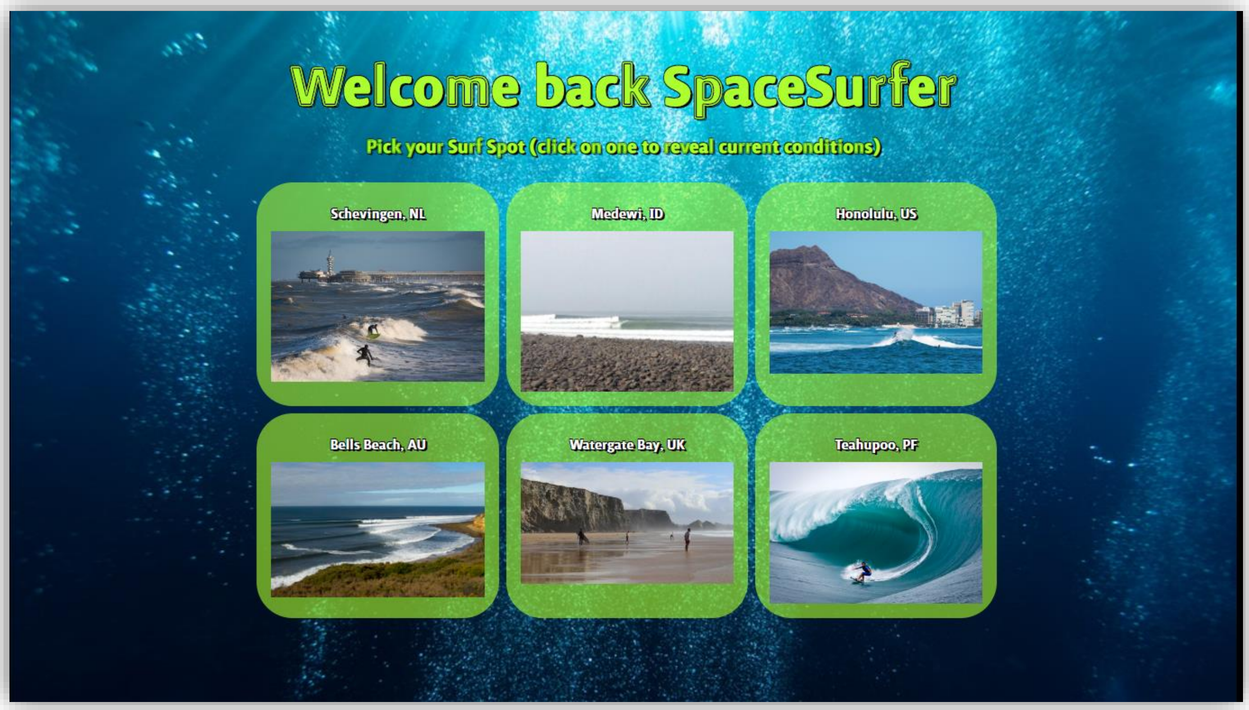
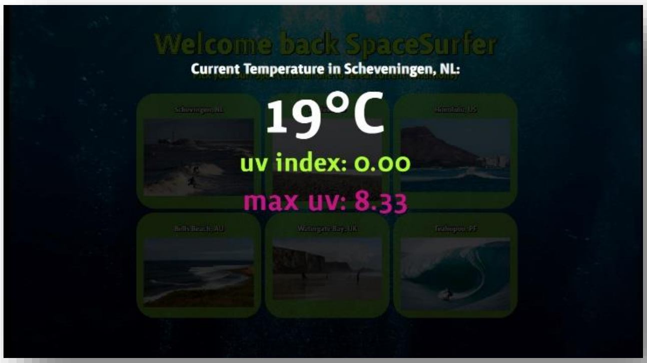
Challenge 3 +Reflection

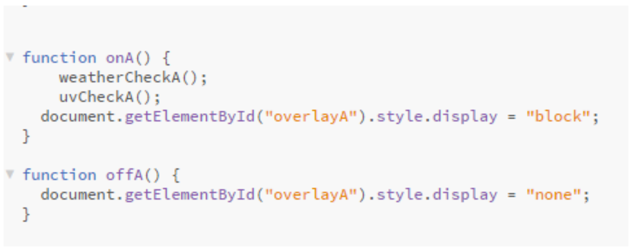
For this last challenge we needed to create a webpage with at least 2 API's included.
I started off by thinking up a theme. For inspiration, I browsed the internet for Public API's to see the possibilities. I settled on a 'surf spot' theme, because I found API's for weather input, wave height, uv and I reckoned I'd make it a bit my own this way. After this I got the sketchbook out and started sketching how I wanted my page to look like. I was going to include the following spots: Scheveningen, Medewi, Honolulu, Bells Beach, Watergate Bay and Teahupoo.
All great surfspots, but also perfect landing spots for the spaceship to land!





When the sketch was done I first downloaded a new coding application that my friend recommended me, called 'Brackets'. I loved it when he showed me how powerful it was. The best feature in my opinion, is the live screen feature, which opens a new browser window that displays your code and changes to live input. So you can see what you do while coding. This made especially the CSS more easy, since this program also highlight the margin and padding areas.

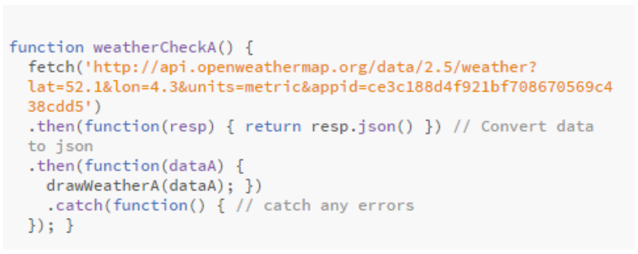
I decided to start making the outline of the boxes how I want them and make Div's that would act as placeholders for the content and lining them out in CSS. After doing that, I realized I needed a space to put the information the user would request when clicking on one of the spots. To do this, I had to Google for a way to make an overlaying Div. What turns out is that there is an option in CSS to display block or none. These options basically tell the div to be displayed or not. Next up, I animated these in JavaScript using onclick.



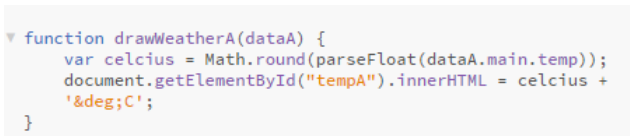
```
function onA() {
    weatherCheckA();
    uvCheckA();
  document.getElementById("overlayA").style.display = "block";
}

function offA() {
  document.getElementById("overlayA").style.display = "none";
}
```

After inserting all the text and images into it's placeholders, it was time to add an API connection to retrieve actual data about these places. I used two API's called: "Open Weather Map" and "Open UV". Both of them required a key. At first I tried to find an API that didn't require one, but most API's have one these days and have data-plans to sell their data. Fortunately, most these keys are free (up to 50 retrieves a day). Both API's required me to make a list of all the coordinates (Lat, Long) in a text file that I could add to the retrieve links.

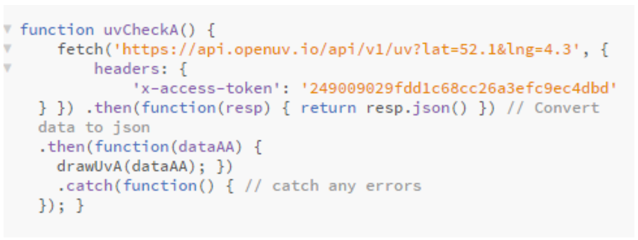
These API's work as follows: You get the specific retrieve link for the data set you choose from the API's documentation website, then you have your code (JavaScript) add the personal API key to the link, then insert the coordinates for the location you want the data from and then retrieve. Next you'll need to make sure that the API gives a response and that it responds with a JSON. Then the JSON file needs to be parched in order for the computer to read. After this you're ready to summon specific data from this dataset anywhere else in the code. You summon It by linking it in JavaScript to a .getElementById in the .innerHTML and the data is ready to be implemented in the HTML.



```
function weatherCheckA() {
  fetch('http://api.openweathermap.org/data/2.5/weather?
  lat=52.1&lon=4.3&units=metric&appid=ce3c188d4f921bf708670569c4
  38cdd5')
  .then(function(resp) { return resp.json() }) // Convert data
  to json
  .then(function(dataA) {
    drawWeatherA(dataA); })
    .catch(function() { // catch any errors
  }); }
```



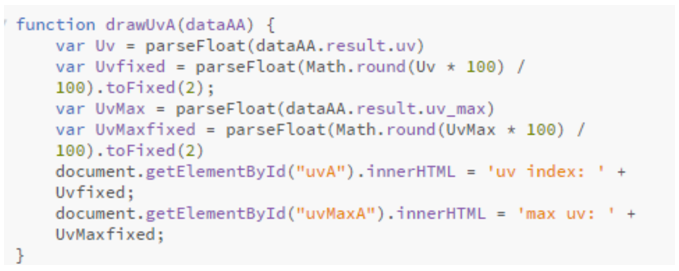
```
function drawWeatherA(dataA) {
    var celcius = Math.round(parseFloat(dataA.main.temp));
    document.getElementById("tempA").innerHTML = celcius +
    '&deg;C';
}
```



```
function uvCheckA() {
    fetch('https://api.openuv.io/api/v1/uv?lat=52.1&lng=4.3', {
        headers: {
            'x-access-token': '249009029fdd1c68cc26a3efc9ec4dbd'
  } }) .then(function(resp) { return resp.json() }) // Convert
  data to json
  .then(function(dataAA) {
    drawUvA(dataAA); })
    .catch(function() { // catch any errors
  }); }
```

For my page, I first did what is written above for only the first one: Scheveningen. I wanted my page to be able to display the current temperature at the spot, the current UV index and the maximum UV index. This would show the mars traveler whether or not he will not get sunburned when landing back on earth.

After completing the code for one working tile (Scheveningen), there must have been a more elegant way of doing what I did. However, since I did not see another way and wanted to make sure it worked, I used the following piece of logic: If one works, 6 will work. So I copied every element that made the Scheveningen tile work and show the data 6 times… And it worked… Sort off…



```
function drawUvA(dataAA) {
    var Uv = parseFloat(dataAA.result.uv)
    var Uvfixed = parseFloat(Math.round(Uv * 100) /
    100).toFixed(2);
    var UvMax = parseFloat(dataAA.result.uv_max)
    var UvMaxfixed = parseFloat(Math.round(UvMax * 100) /
    100).toFixed(2)
    document.getElementById("uvA").innerHTML = 'uv index: ' +
    Uvfixed;
    document.getElementById("uvMaxA").innerHTML = 'max uv: ' +
    UvMaxfixed;
}
```

In the end, I decided to have a 'Ocean Man' play on the background of the page to get the SpaceSurfer in the right vibe to land his spaceship! ~~*Sorry*~~

Next time, I will definitely find a more elegant way to make a lot of repetitions and spend more time on formatting my code. I think my code is still quite readable, only less repetition is the goal!

Result can be found here: https://julesb98.github.io/Challenge_3_JulesB/