http://titan.dcs.bbk.ac.uk/~zgeorg01/wd/fma/task2/apiweather.html

```
const CONFIG = {
    API: {
        URL: "https://api.openweathermap.org/data/2.5/weather?",
        KEY: "0d656e1048bce2869ea884fd96954f99",
        PARAMS: "units=metric",
        IMAGE_URL: "https://openweathermap.org/img/w/",
    },
    CARDINALS: ["Northerly", "Nort-Easterly", "Easterly", "South-Easterly",
"Southerly", "South-Westerly", "Westerly", "North-Westerly"],
    MEDIA: {
        FOLDER: "media",
    },
    LOCALE: "en-UK",
    CONST: {
        MPS_TO_KMPH: 3.6,
        MPS_TO_MPH: 2.23694
    },
    WARNINGS: {
        TEMP_ABOVE: 35,
        TEMP_BELOW: -5,
        WIND_ABOVE: 50
    },
}
```

With the API object we can change the openweather url, access key, the parametres and image url. Cardinals represented as an array object will help us to store mostly used directions so that the function can iterate over them and display the current wind direction. Media folder will specify the folder in which all media/images file will be stored. Locale will be used to specify the date formatting . Const will contain all constants needed for the calculating functions. Warnings will be used to specify the boundary conditions for the severe messages, as per the requiremets. It can also help us to debug/test the module wether the severe images are properly shown.

```
/**
 * Converts Celsius to Fahrenheit and return it
 * @param {Number} Celsius
 * @returns {Number}
 */
const fahrenheit = celsius => Math.ceil(celsius * 1.8 + 32)
```

Arrow functions introduced in ES6 are a great shortcut for recursive functions or online calculations. In our case, the function expects a parameter **celsius** that can be any integer number to convert the Celsius in Fahrenheit using the formula from https://www.mathsisfun.com/temperature-conversion.html.

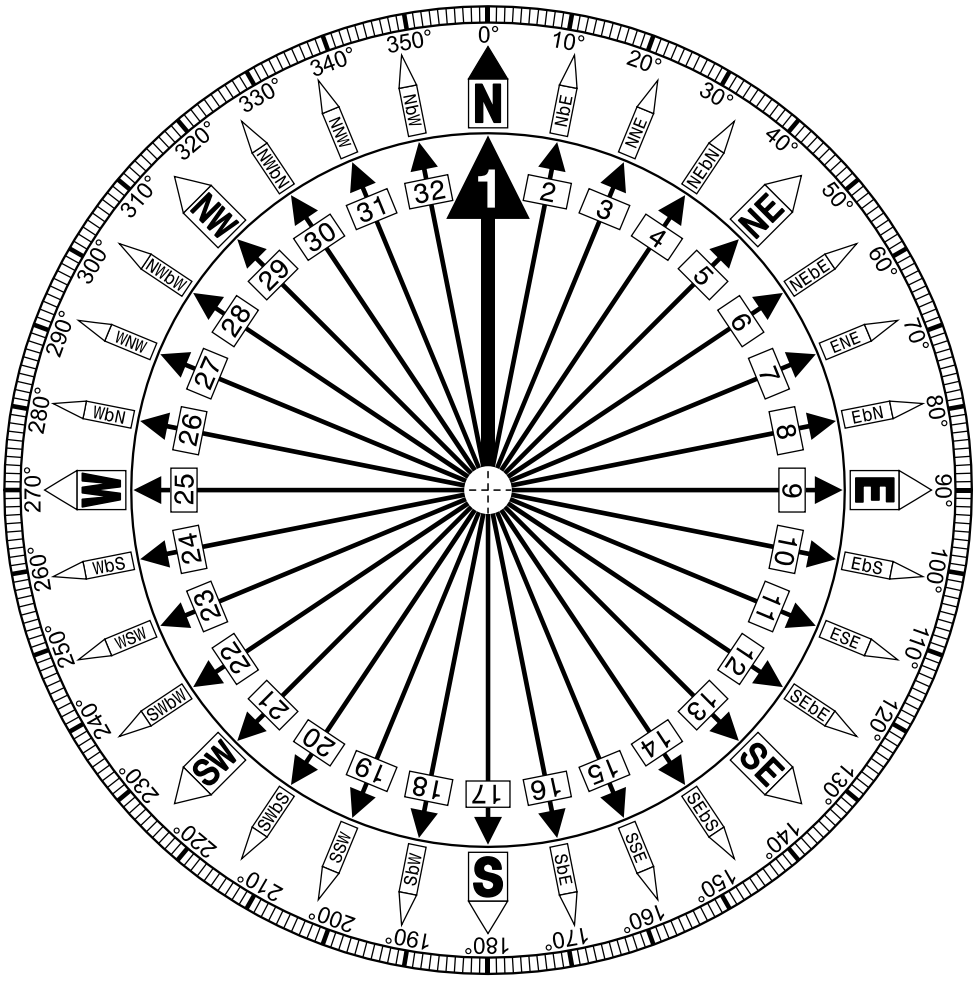
The result of the calculation is in float point notation but I have noticed that google weather API rounds the number up to the next largest integer (for instance 13 Celsius are calculated to 56 Fahrenheit, where it is actually, 55.4) so that I use **Math.ceil** to do the same.

```
/**
 * Converts degrees in cardinal text
 * @param {Object} degree
 * @returns {String}
 */
const degree_to_cardinal = degree => CONFIG.CARDINALS[Math.round(degree /
(360 / CONFIG.CARDINALS.length)) % CONFIG.CARDINALS.length]
```

As per the requirements we have to convert the wind degree to cardinals to display it as a text. I found some ideas how to do that here https://stackoverflow.com/questions/64709244/shell-script-to-convert-degrees-to-cardinal-direction and understood that array/list representation of the directions will be needed.

```
CONFIG.CARDINALS
[Math.round(degree / (360 / CONFIG.CARDINALS.length)) % CONFIG.CARDINALS.length]
```

Will divide the degree to the result of one turn devided to all cardinals. In our case it will be the length of the array that contains all directions they are just 8 so 360/8 = 47.5 to get the degree of each direction. Then by dividing the degree by that number for example 120/47.5 we get 2.52, since we do not need a whole number we round it to get 3 and then check what will be the reminder of devision 3/8 which is 3. So 3 is our index that will get South-Easterly from the array which is abut to be correct since 120 degrees are more closer to SE than E. Ofcourse if we add more directions we can get more precise result by only adding more cardinals in the array (the idea of having config pays out) and then without touching the actual function it should work properly as long as we add the cardinals properly (although some rounding adjustments may be needed for precision, not tested). But I think that it is out of the scope of the task requirements and I did not play with that further.



```
/**
 * Converts the wind direction to display wind sign
 * @param {Number} wind
 * @returns {Number}
 */
const wind_direction = wind => wind > 180 ? wind - 180 : wind + 180
```

I was inspired by the API https://openweathermap.org/city/2643743 that shows a small wind image that shows the actual wind direction and decided to implement it in the same way into my solution. So used the browser dev tools to see that they use css transform rotate to specify the rotation of the image. The next question was to calculate the degree to rotate properly and by reading https://old.oceanrowing.com/weather/understanding_wind_direction.htm I understood how they calculate it and was able to implement my solution. It is simple check if the wind degree is bigger than 180 we substract 180 to get the direction that the wind is moving towards and add 180 if the wind degree is less than 180.

```
/**
 * Converts speed from metres per second to kilometres per hour
 * @param {Number} speed
 * @returns {Number}
 */
```

```
const wind_to_kph = speed => (speed * CONFIG.CONST.MPS_TO_KMPH).toFixed(1)

/**
 * Converts speed from metres per second to miles per hour
 * @param {Number} speed
 * @returns {Number}
 */
const wind_to_mph = speed => (speed * CONFIG.CONST.MPS_TO_MPH).toFixed(1)
```

Both functions just convert wind speed, which is in metres per second from the API to miles per hour and kilometres per hour. In principle these two function may be redundant as they do same thing and can be done directly in the template/ html but I decided to have them separately than do calculations inside the html.

```
/**
 * Converts speed from meters per second to miles per hour
 * @param {Number} timestamp
 * @param {Number} type -Two options 0 or 1 are available. Default is 0 and
only the date will be returned, where 1 will return only the time
 * @returns {String}
 */
const format_date = (timestamp, type = 0) => new Date(timestamp *
1000).toLocaleString(CONFIG.LOCALE).split(",")[type].replaceAll("/", "-")
```

There is a built in function toLocaleString in JavaScript https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toLocaleString that returns a date based on passed locale as an argument, I decided to use it so that could give me more flexibility and code reuse. Because the format of the returned value is like ***08/07/2021, 00:04:43*** I had to use only the date and then replace all foreslashes with dash as per the FMA task requirements. Prior to improve code reuse I added optional type argument because it is needed for displaying the sunset and sunrise time.

```
/**
 * Checks the time zone value and it's sign and composite with the time
difference (in hours) for the chosen location
 * @param {Object} city
 * @returns {String}
 */
const calc_zones = city => `GMT${Math.sign(city.timezone) === 1 ? " +": " -
"}` + Math.abs(city.timezone / 3600) + ":00"
```

Folloing the Openweather API I wanted to display the GMT since it is provided within the JSON response, although the format is either positive or negative integer (seconds). Because we have to show the sign we check whether it is positive or negative and return + or – sign, then concatenate the timezone/3600 to get the hours and lastly concatenate the remaining seconds (for some reason I did not find GMT with different secods other than 00 so left it static, but that may be wrong)

```
/**
 * Combine different images for low temperature, high temperature and strong
wind anomalies based on the warning parameters from the config
 * @param {Object} city
 * @returns {String}
 */
function severe_image(city) {
    let image = ""
    let temp = Math.round(city.main.temp)
    if (wind_to_mph(city.wind.speed) > CONFIG.WARNINGS.WIND_ABOVE)
        image += `<img src="${CONFIG.MEDIA.FOLDER}/strong-wind.png">`
    if (temp > CONFIG.WARNINGS.TEMP_ABOVE)
        image += `<img src="${CONFIG.MEDIA.FOLDER}/high-temp.png">`
    else if (temp < CONFIG.WARNINGS.TEMP_BELOW)
        image += `<img src="${CONFIG.MEDIA.FOLDER}/low-temp.png">`
    return image
}
```

The function accept city coming from the JSON object and checks the city wind speed against those in the Config and adds severe wind warning as an image if the wind is stronger. Same applies to the temperatures above or below the normal set in the main config. Each severe image will be added one after another if the temperature and wind are in dangerous value.

```
/**
 * Makes AJAX request to the weather API and trying to fetch the JSON Object.
 * Passes the JSON data to the template to be rendered if the response code is 200/OK or
display alert message otherways
 * @param {JSON Object} city
 */
function fetch_data(city){
    fetch(CONFIG.API.URL + CONFIG.API.PARAMS + `&q=${city},gb&APPID=` + CONFIG.API.KEY)
    .then(responce => responce.json())
    .then(data => {
        if (data.cod == 200) {
            $("#city-weather").html("")
            $("#city-weather").append(TEMPLATE(data))
        } else {
            alert(`Information for city ${city} can not be found!`)
        }
    })
}
```

The function expects city name to be passed as an argument and then construct the request query using the config url, additional parameters and api key. Using AJAX the constructed request will be sent and the response read, if the cod node in the json is 200 the data from the json will be sent to the template and append to html id #city-weather. In case that the cod is different value than 200 an error message will be displayed.

```
$("body").ready(doc => {
    $("#country").on("change", element => {
        let cities = $("#country :selected").text() == "Northern Ireland" ? "nireland-
cities.html" : $("#country :selected").text().toLowerCase() + "-cities.html"
        fetch(cities).then(
            response => {
                if (response.statusText != "Not Found") {
                    $("#cities").load(cities)
                    $("#cities").on("change", selected => {
                        city = $("#cities :selected").text()
                        fetch_data(city)
                    })
                } else {
                    alert(`The file ${cities} can not be read!`)
                }
            }
        )
    })
})
```

Thinking for the simplest way to iterate over the provided html files containing cities as options. I saw a pattern that repeats in the file naming where the only difference was nireland-cities which contains North Ireland cities. So I implemented the idea using ternary statement

```
let cities = $("#country :selected").text() == "Northern Ireland" ? "nireland-
cities.html" : $("#country :selected").text().toLowerCase() + "-cities.html"
```

that reads the selected value if the value is different than North Ireland I am looking for the file directly since the other files are named properly. In case that North Ireland is selected I change the name to the appropriate to match the file name. After that next block of code is trying to read/access the file and if it exists and the statusText is not “Not found”

```
        fetch(cities).then(
            response => {
                if (response.statusText != "Not Found") {
                    $("#cities").load(cities)
                    $("#cities").on("change", selected => {
                        city = $("#cities :selected").text()
                        fetch_data(city)
                    })
                } else {
                    alert(`The file ${cities} can not be read!`)
                }
            }
        )
```

We load all options from that file that contains the cities for the celected country and load it to css id #cities. Because the city. Then we listen for change event on the selection and if the user selects a city from the options `fetch_data(city)` will be called and the city name will be passed as an argument to the function and then it will send an API request and display the weather.