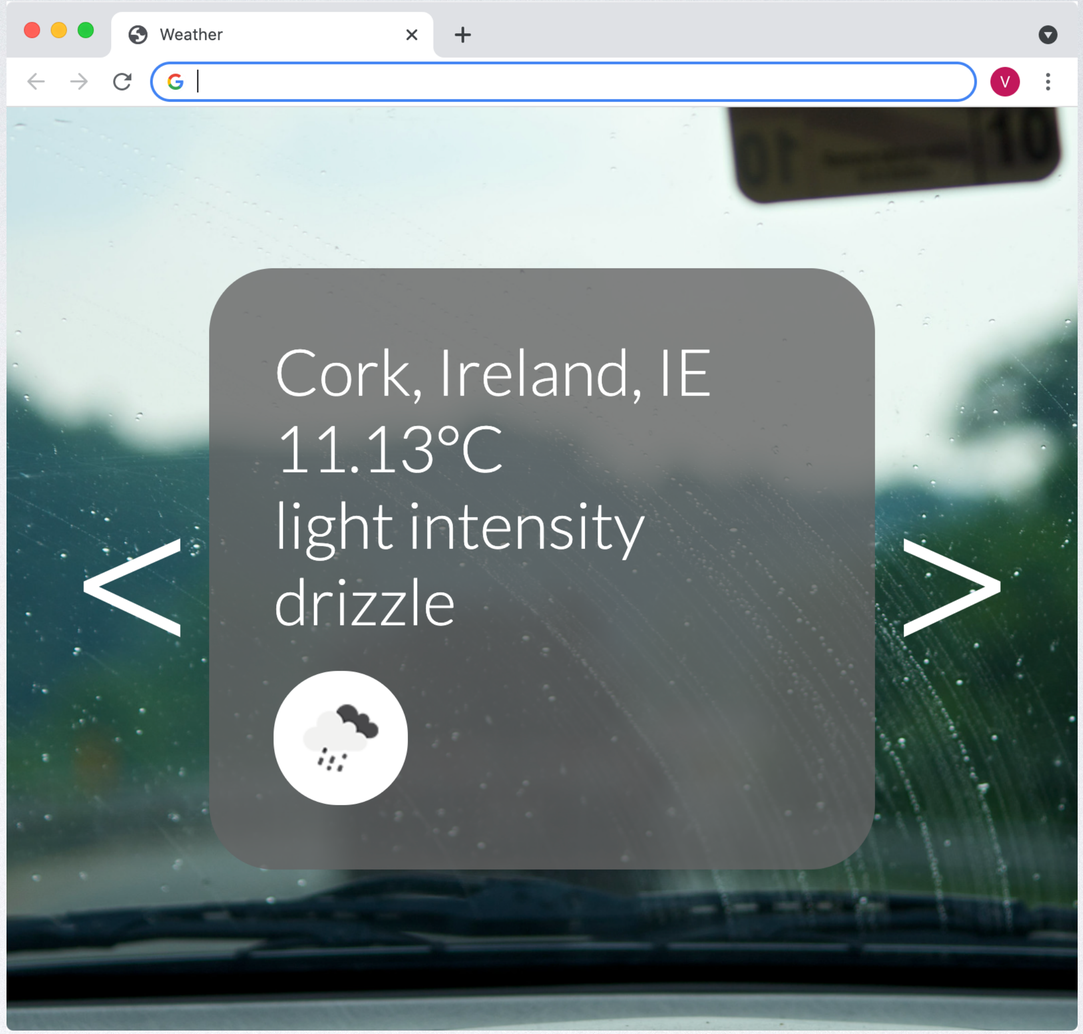
WEATHER API

Part I

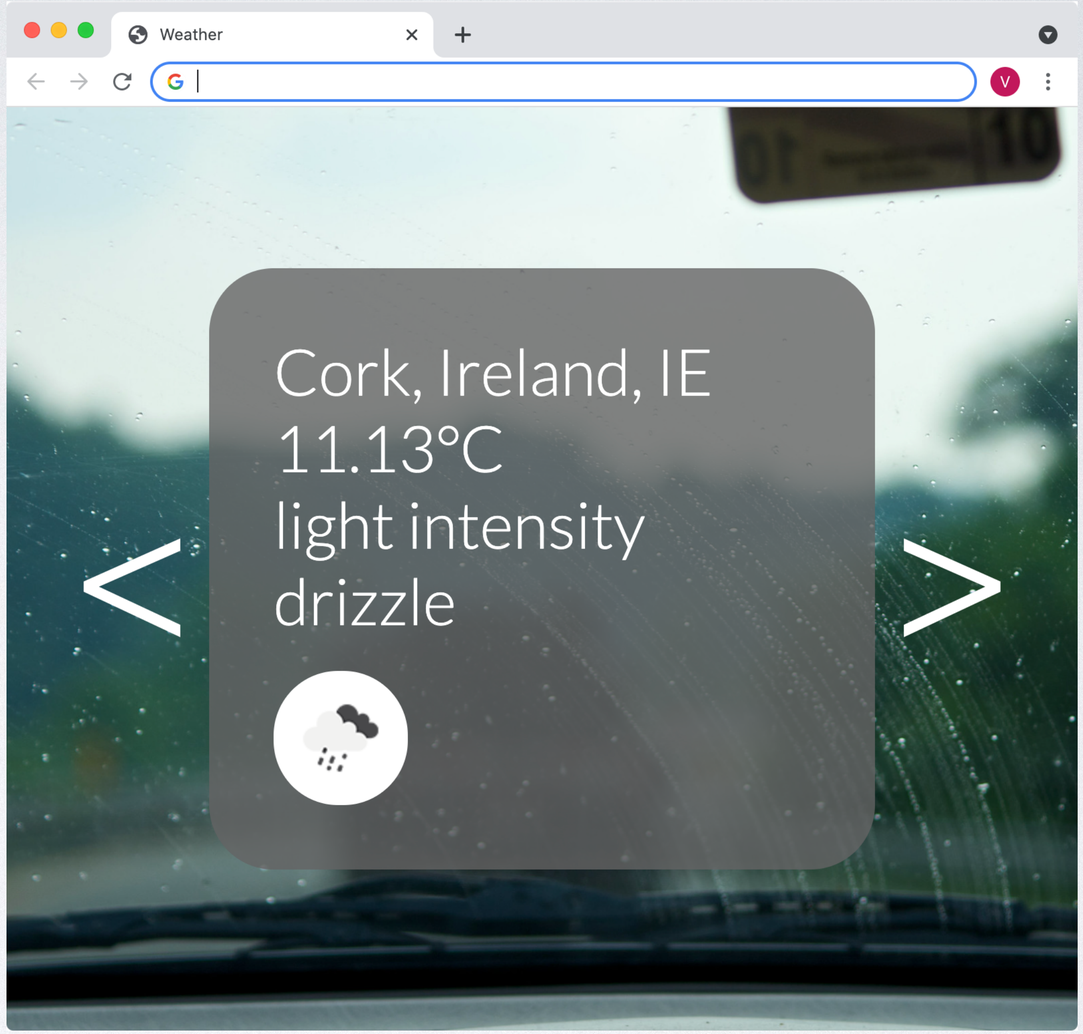
You must create a weather viewer web application.

It retrieves information from a web service to give up-to-date and accurate details of the weather in various cities around the world.

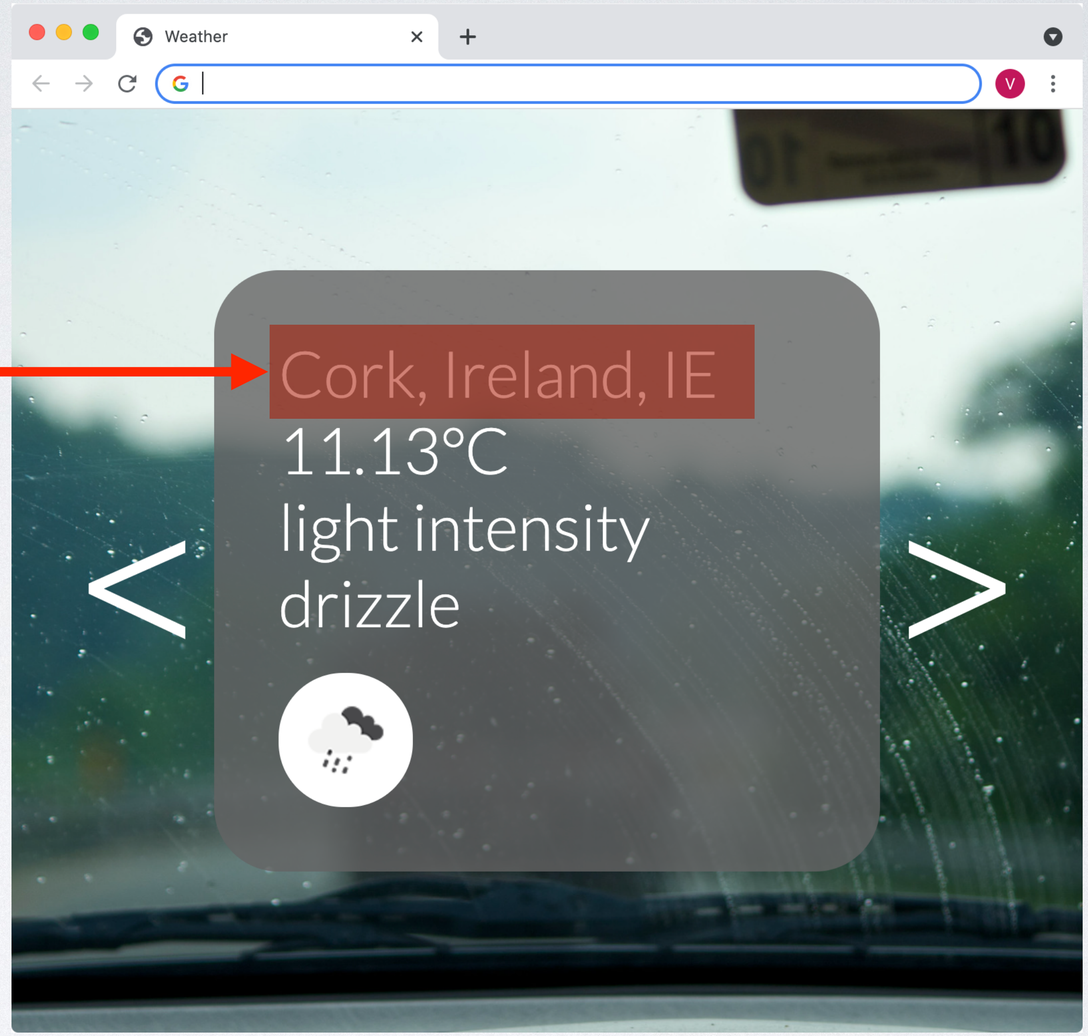


You provide the Name of the City and the web service provides the data.

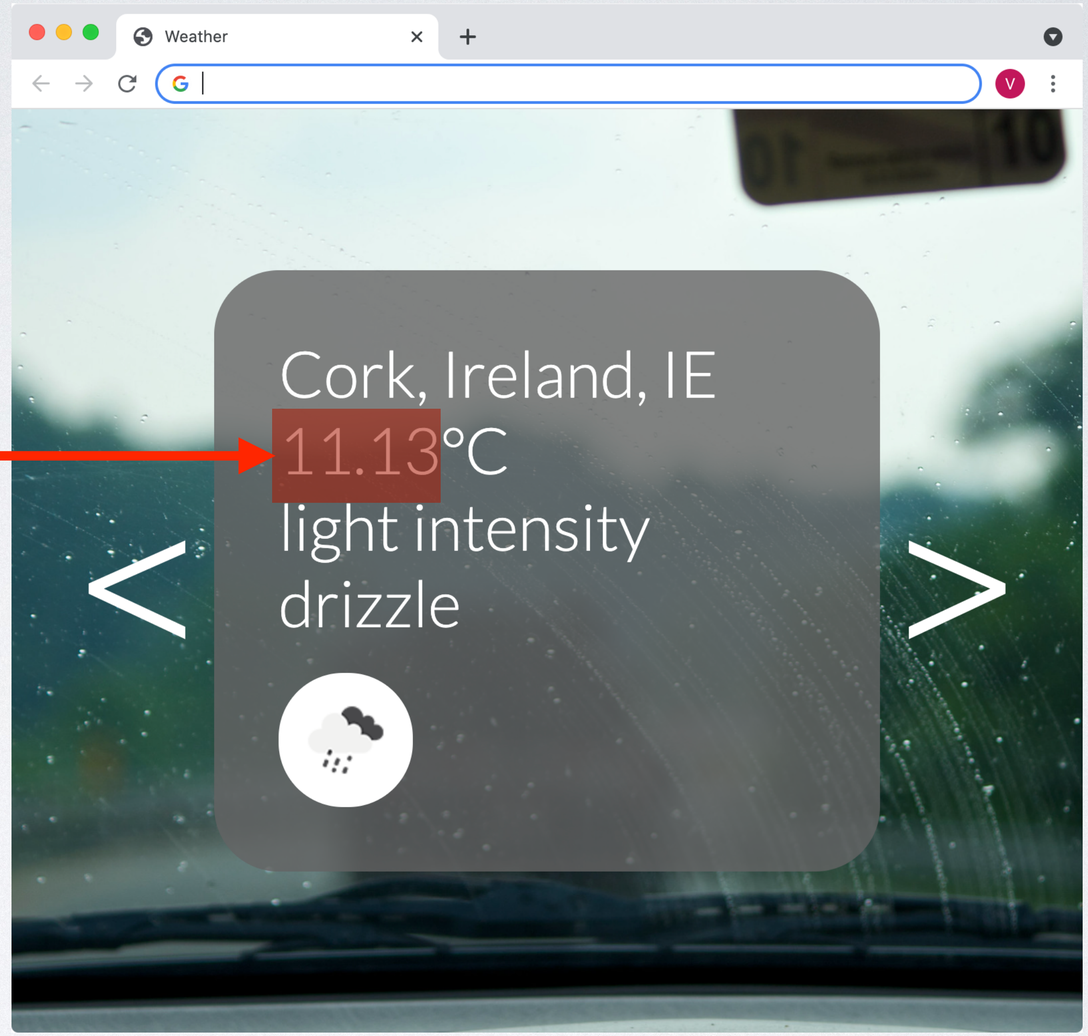
You must design a page similar to the one shown here that displays this data.



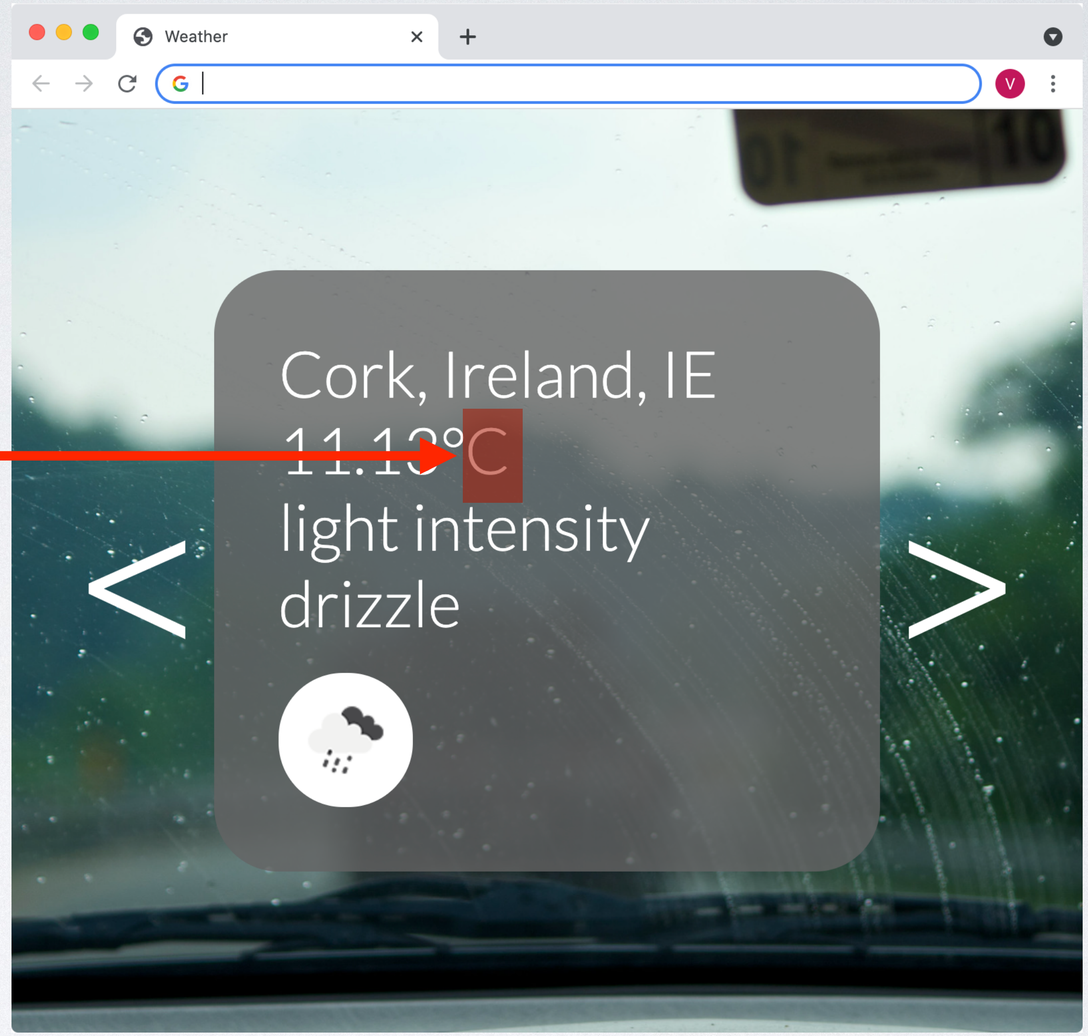
It must have space for the name of the city.



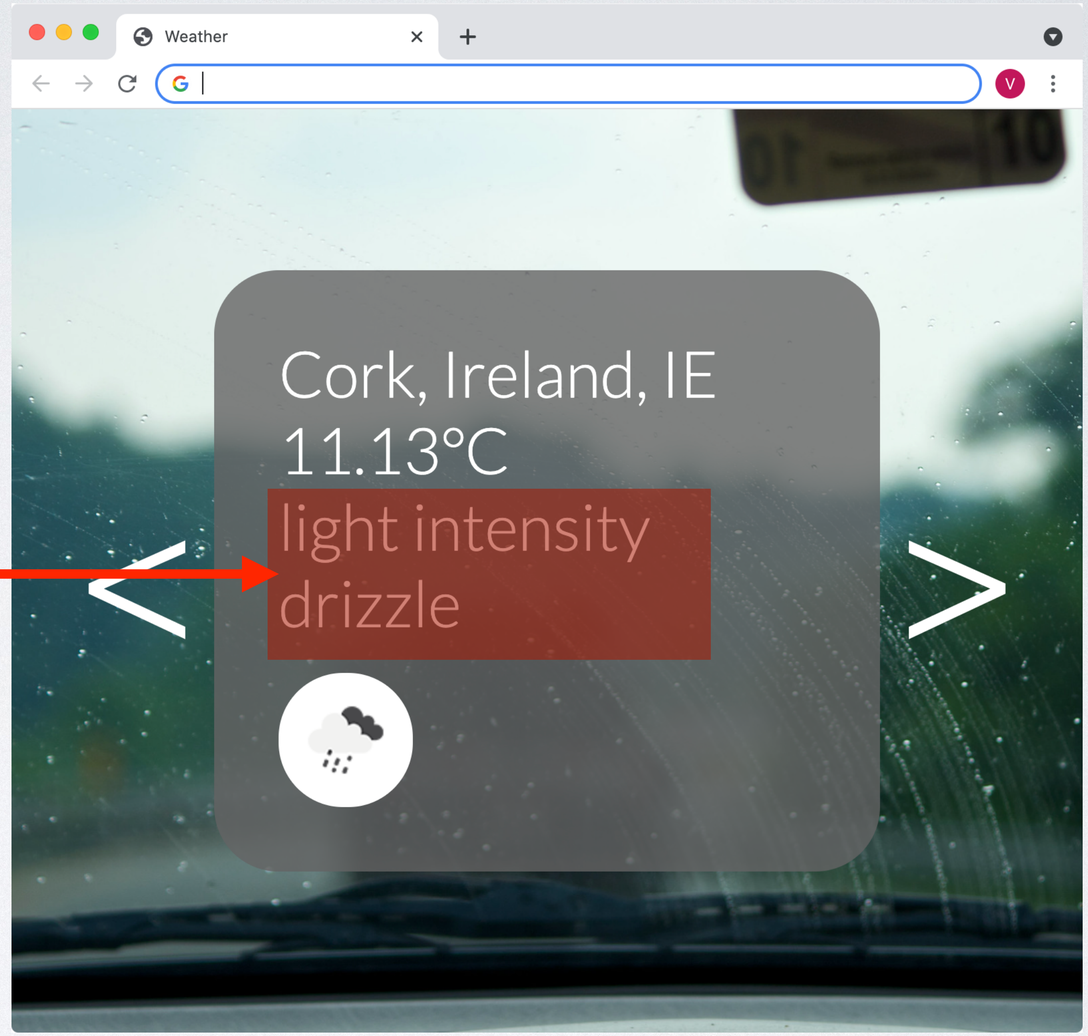
It must have space for the temperature (retrieved from the web service)



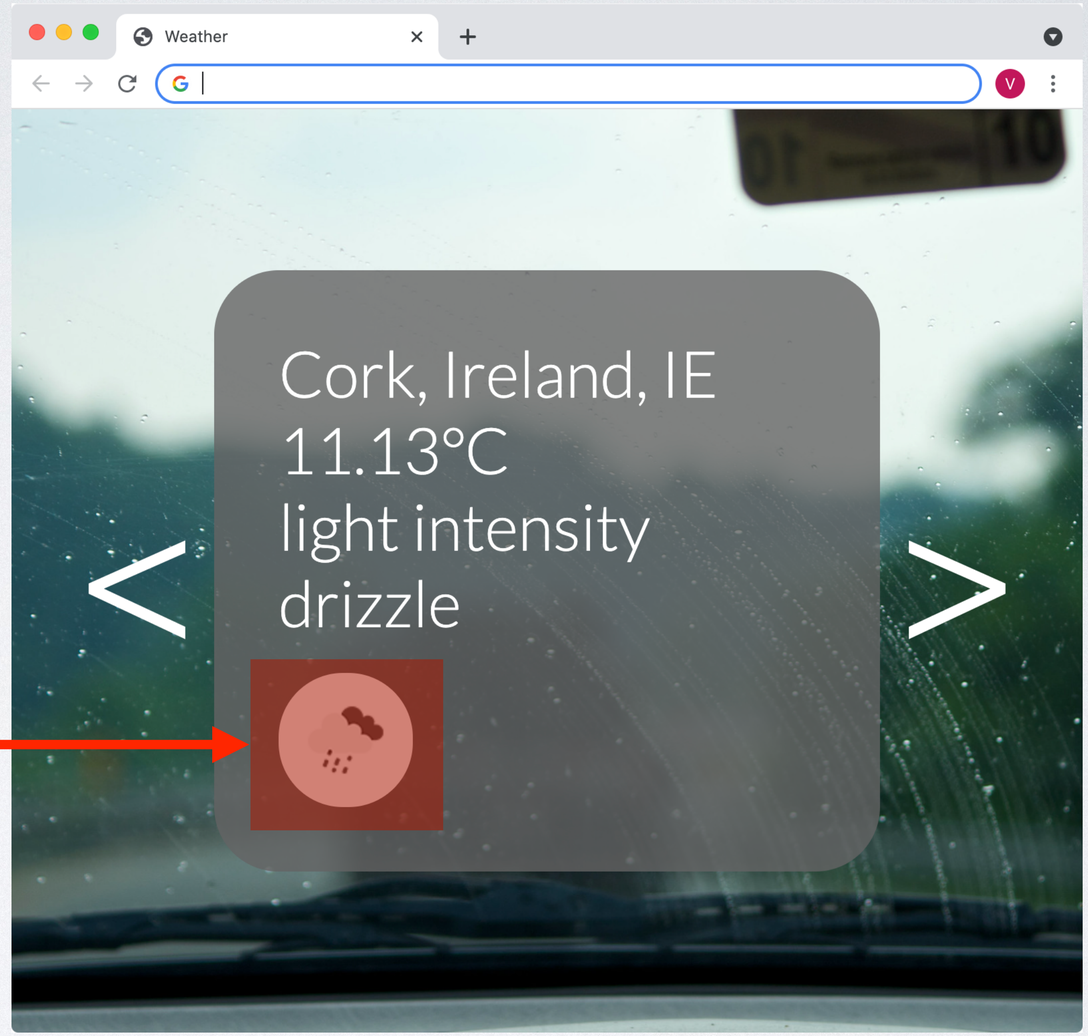
It must indicate the scale being used (Fahrenheit or Centigrade). You specify this when you make your requests to the web service.



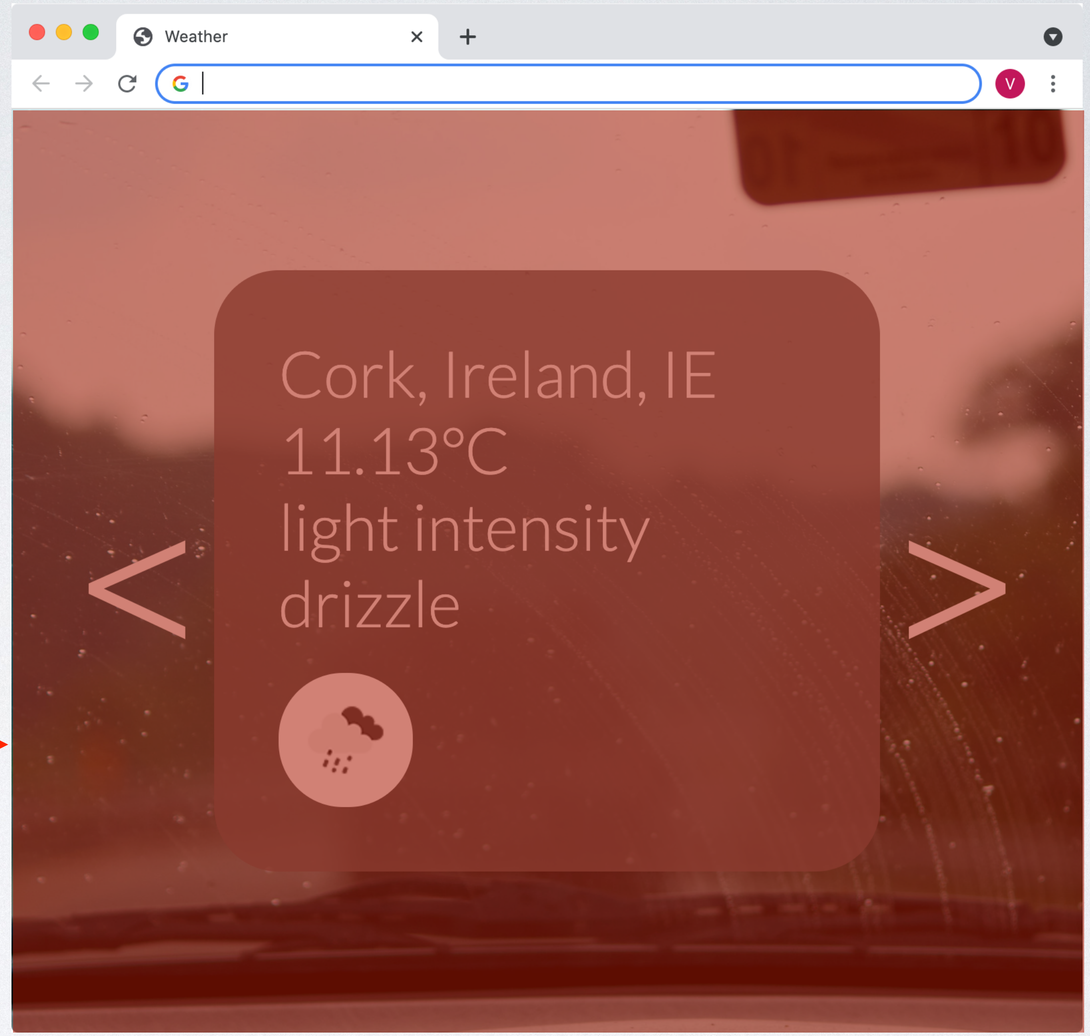
You must show the description of the weather (from the web service)



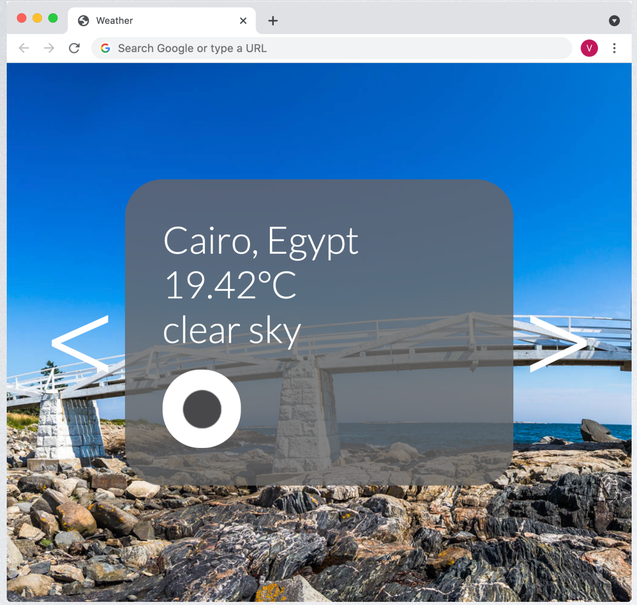
You must show the icon (Information on the icon is returned by the web service).

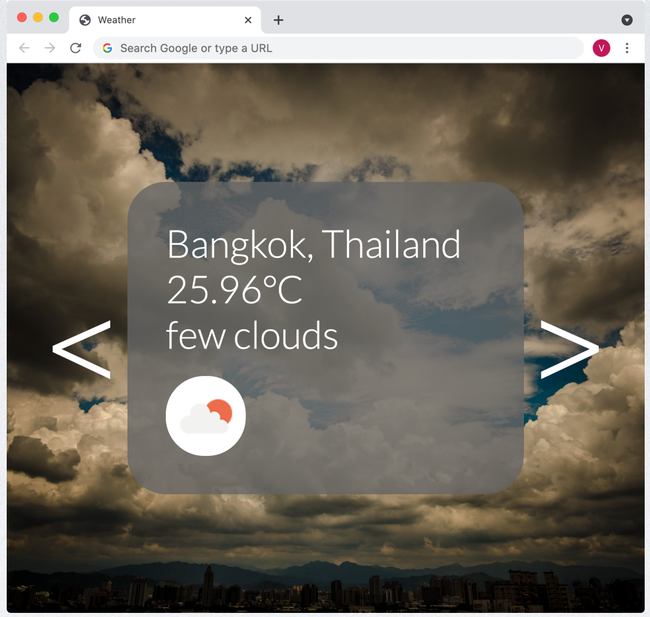


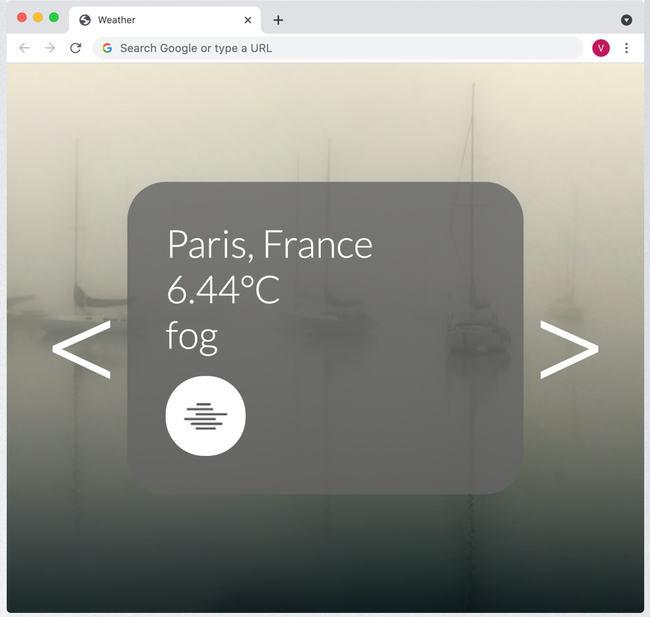
Finally, you must show an image covering the entire background. This image should represent the broad category of weather for the city currently being shown.

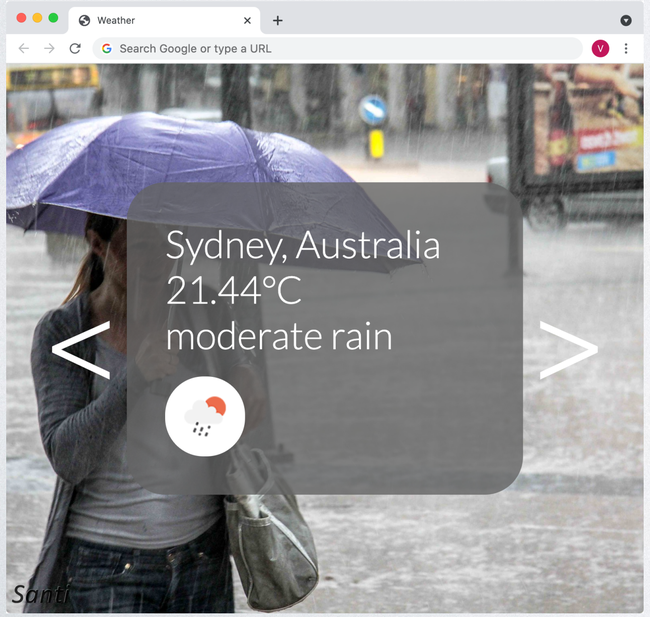


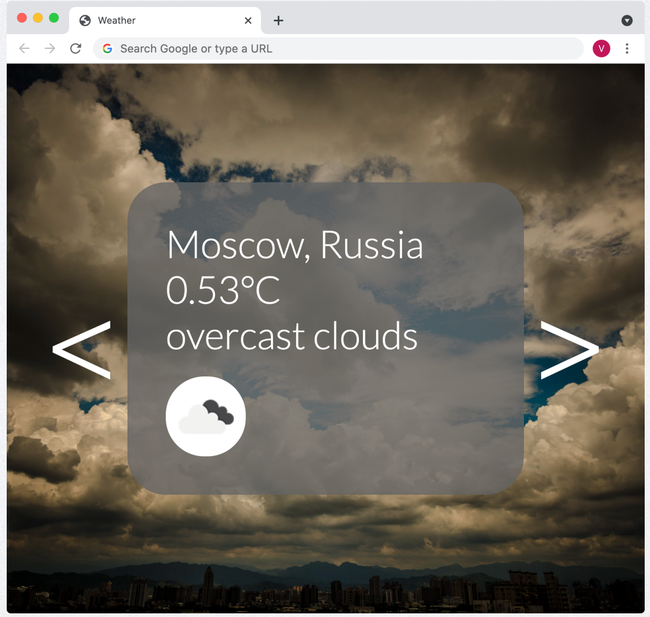
Background Examples

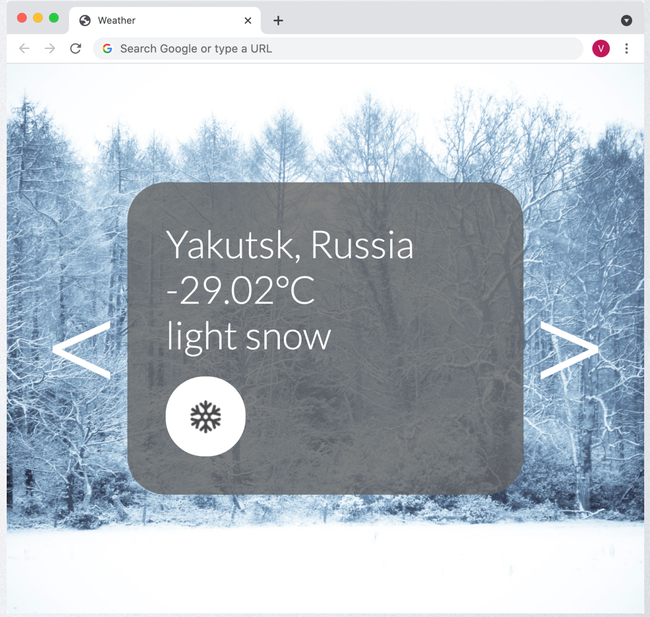






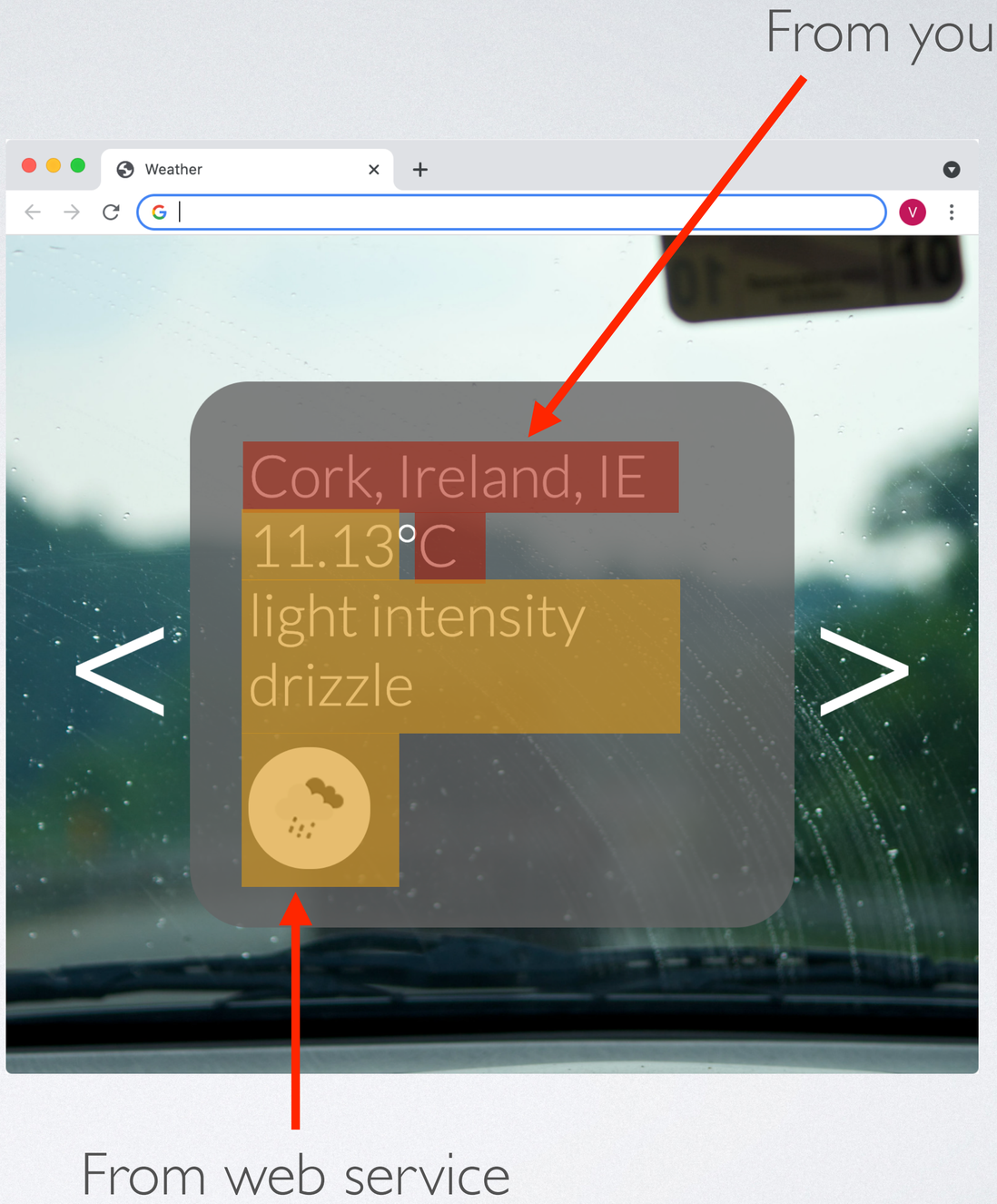






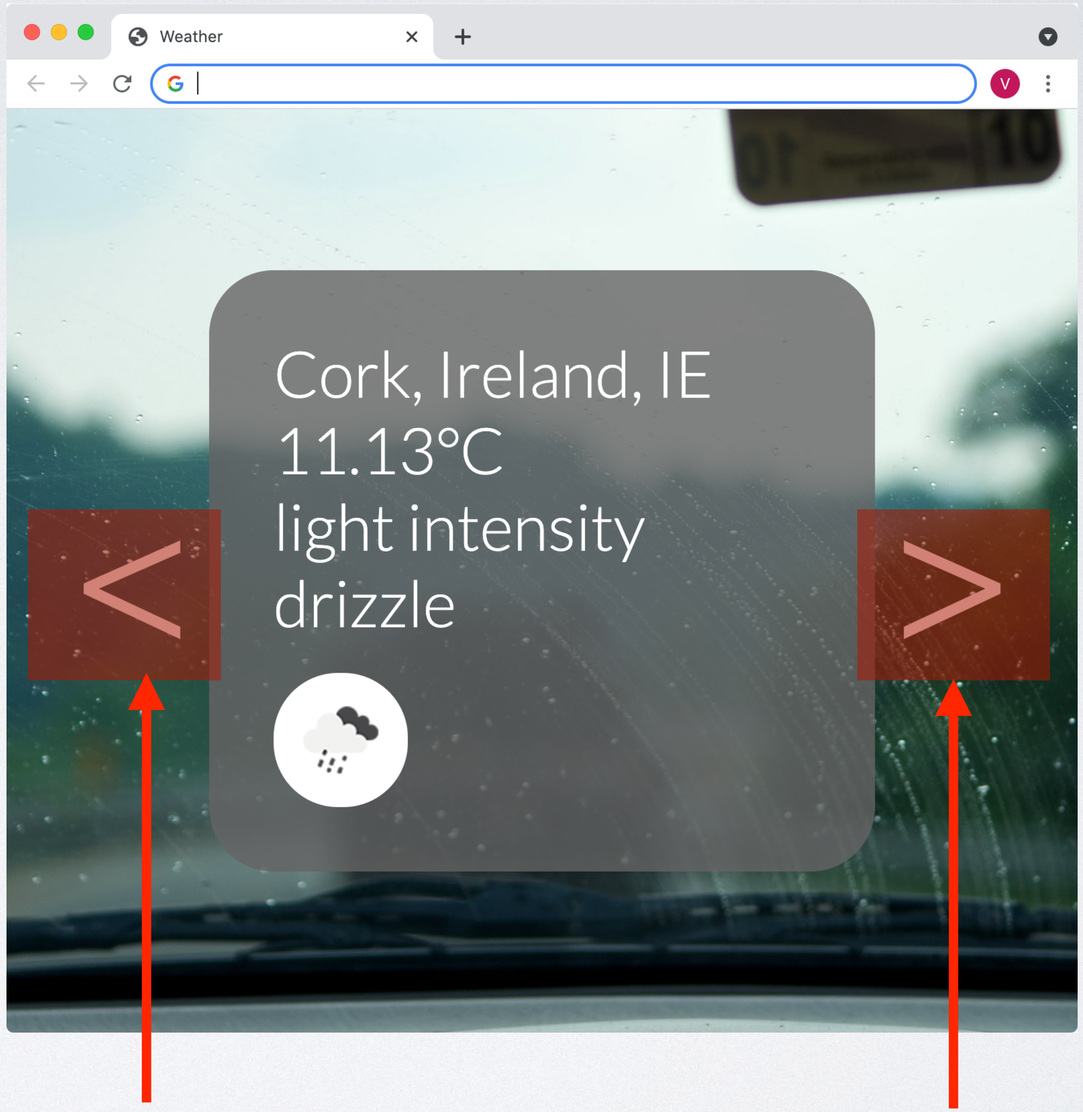
You can source images on sites like Flickr (you will need 7 images).The images you use are up to you but they should represent the 7 weather types (see notes on the API) and be large enough to fill a full page.

Summary: You must send data to the web service (e.g. the city name , and the scale you want the temperature in) and get back (and display) the temperature, description, and icon details.



Part 2

You must allow your web app to cycle through a series of cities. There should be controls onscreen to allow you see the next or previous cities. (the cities should wrap around when they get to either end of the sequence. i.e. after the last city, ***next*** should bring you the first.)



You can use these or your own array of cities. If using your own make sure the API has data for each city you use.

```
var cities =

  [
  "Cork, Ireland, IE",
  "Cairo, Egypt",
  "Moscow, Russia",
  "Toronto, Canada",
  "New York, USA",
  "Hong Kong"  ,
  "Paris, France",
  "Sydney, Australia",
  "Yakutsk, Russia",
  "Bangkok, Thailand"
  ];
```

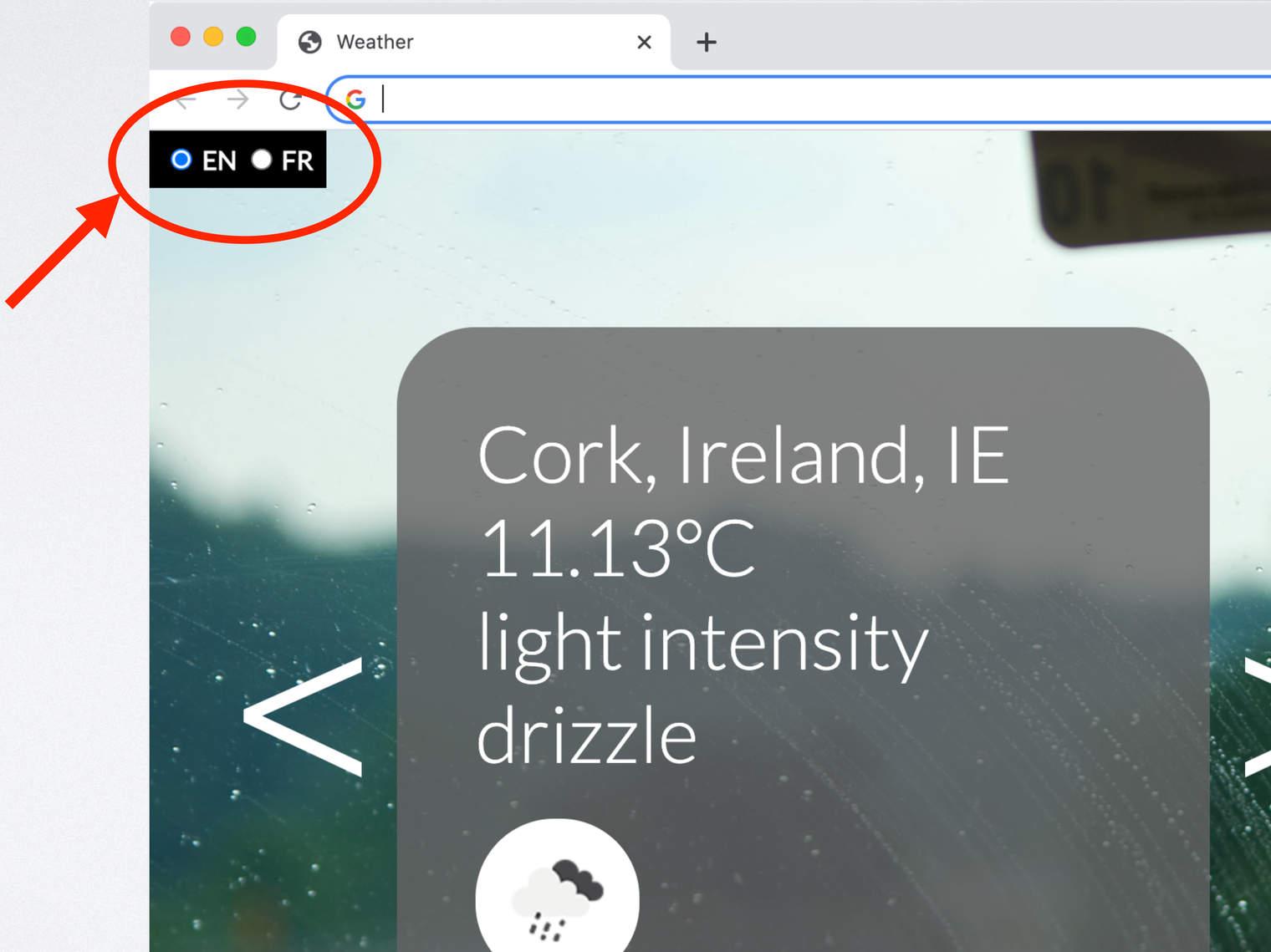
Part 3

Part 3 adds 2 embellishments to the app:

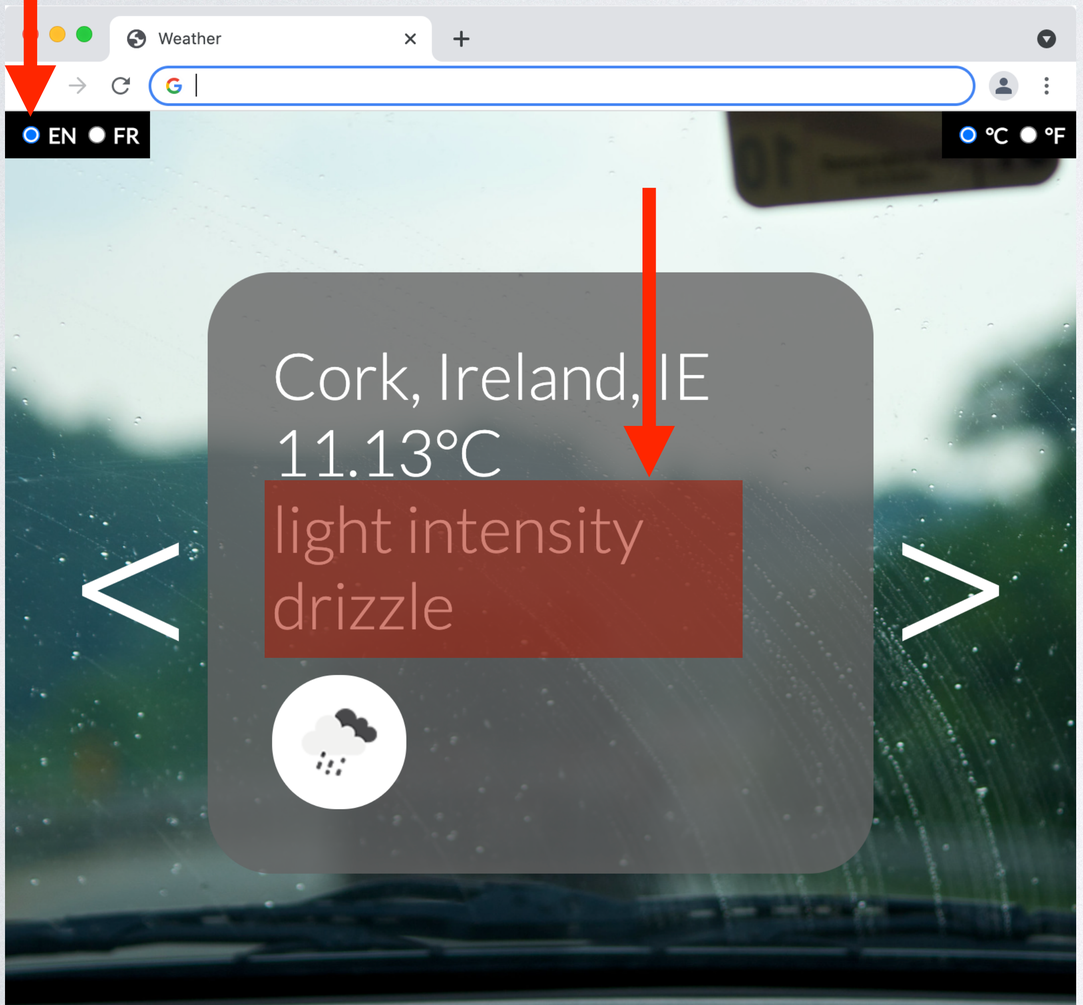
Allow user to specify the language the description will be in.

Allow user to specify temperature scale that will be used.

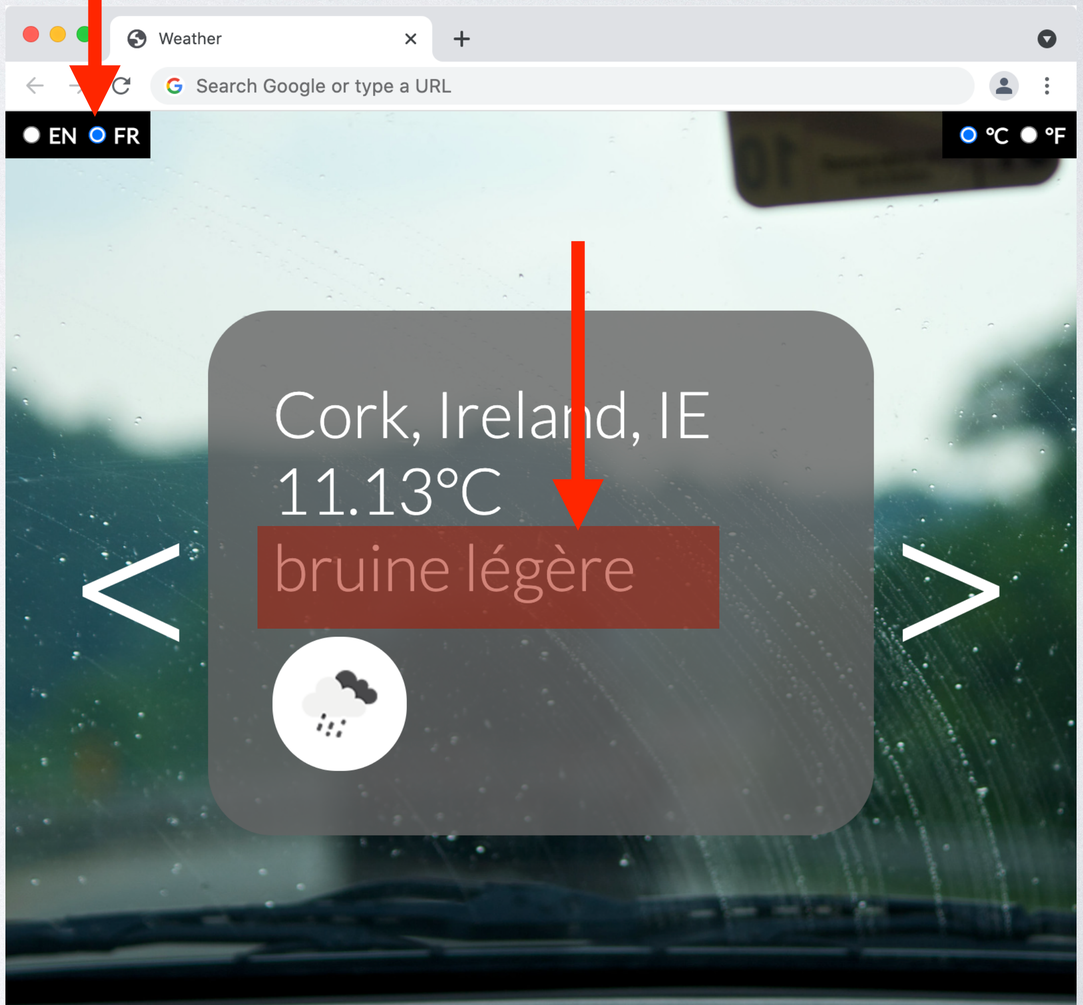
You should be able to choose between English or French for the language of the description.



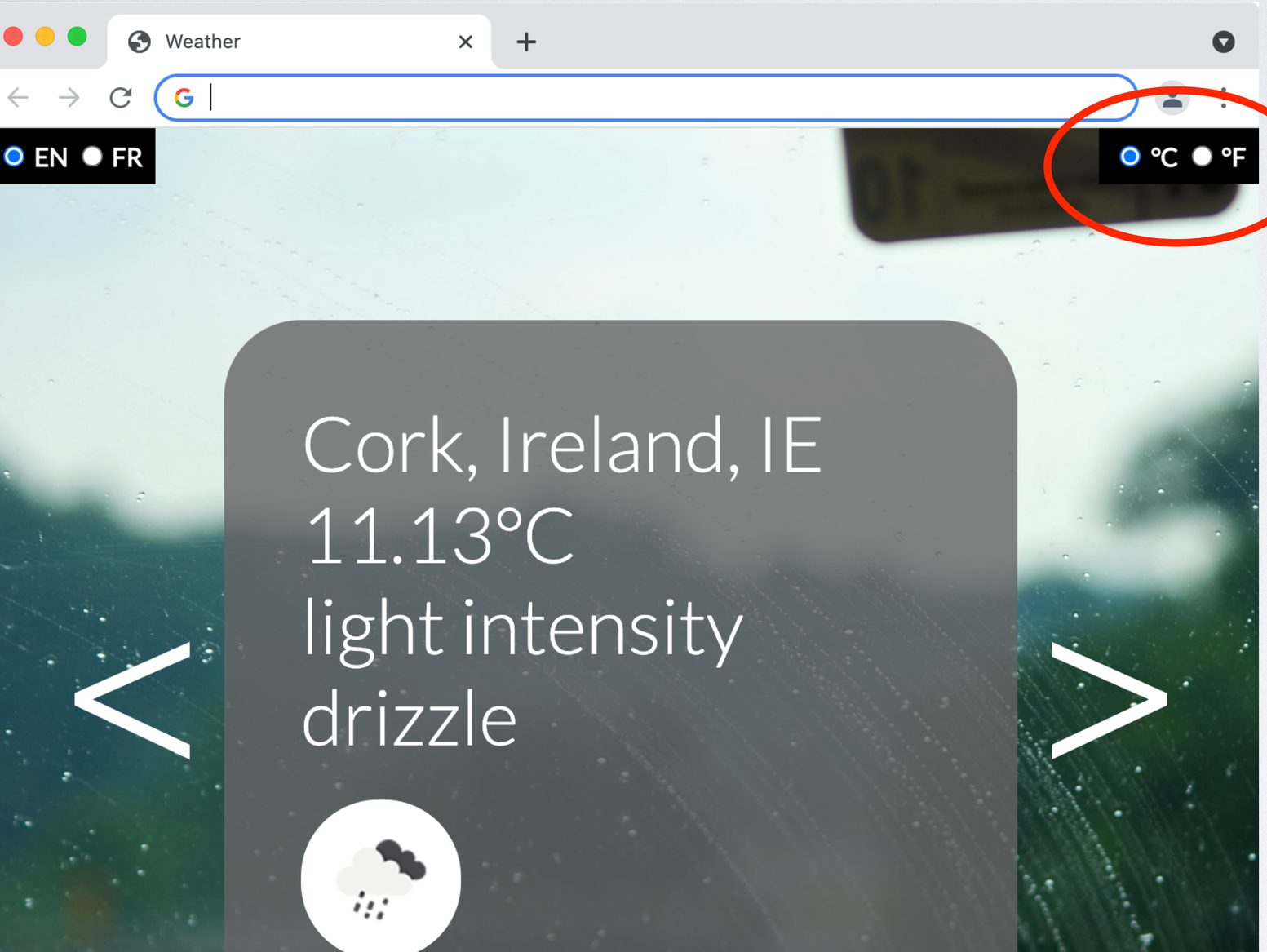
English



French

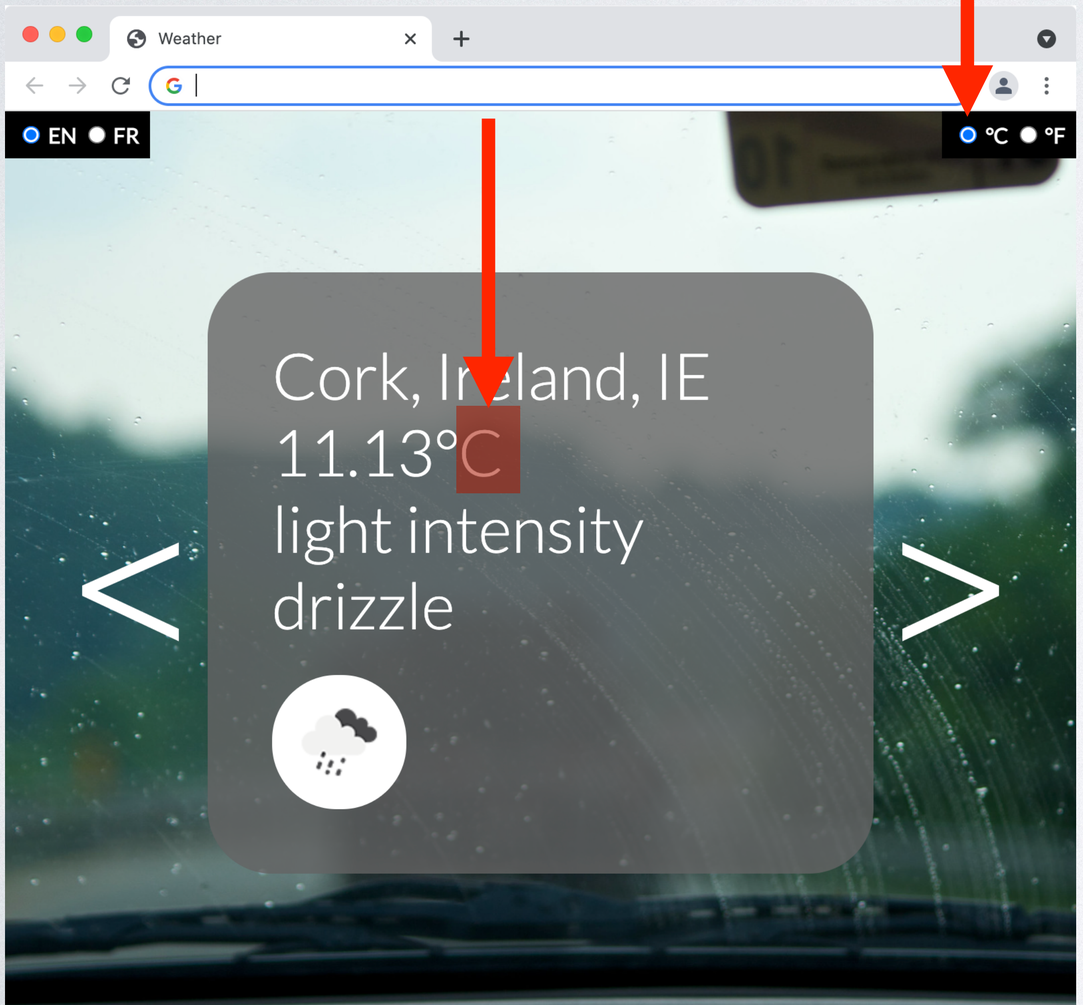


2

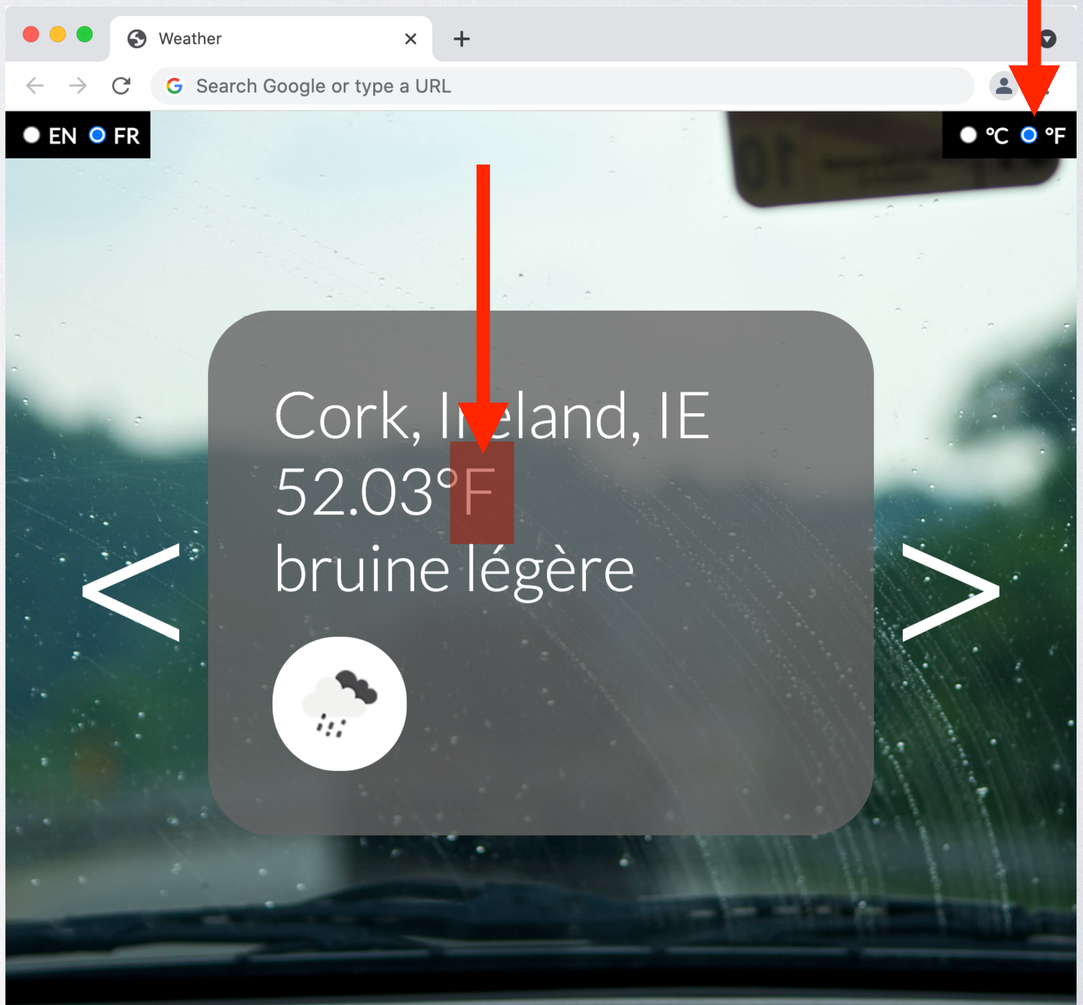


You should be able to choose The temperature scale as well. This information will be sent to the web service.

Centigrade



Fahrenheit

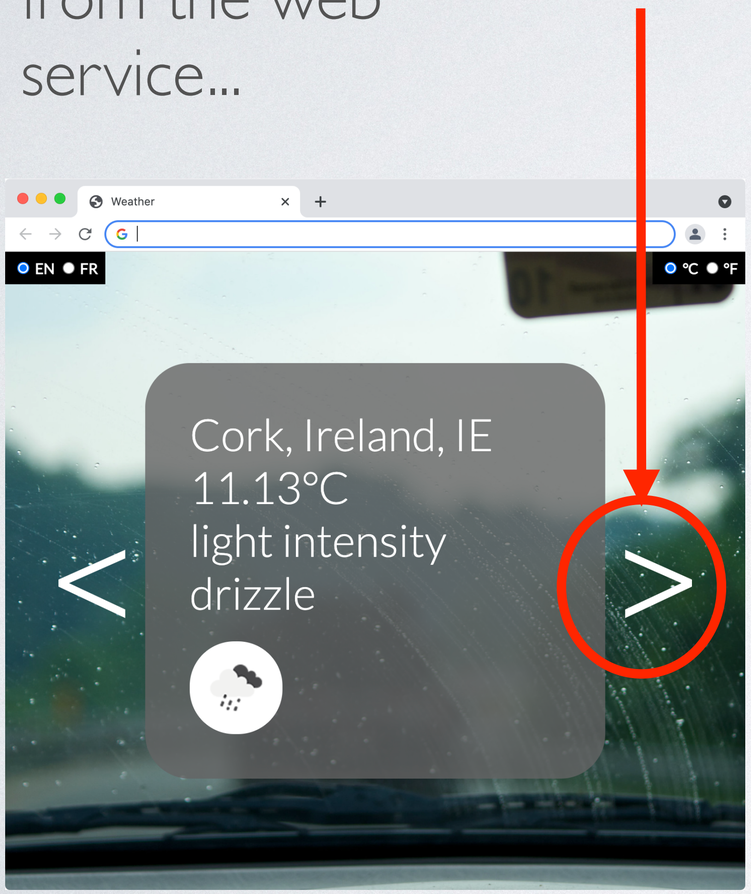


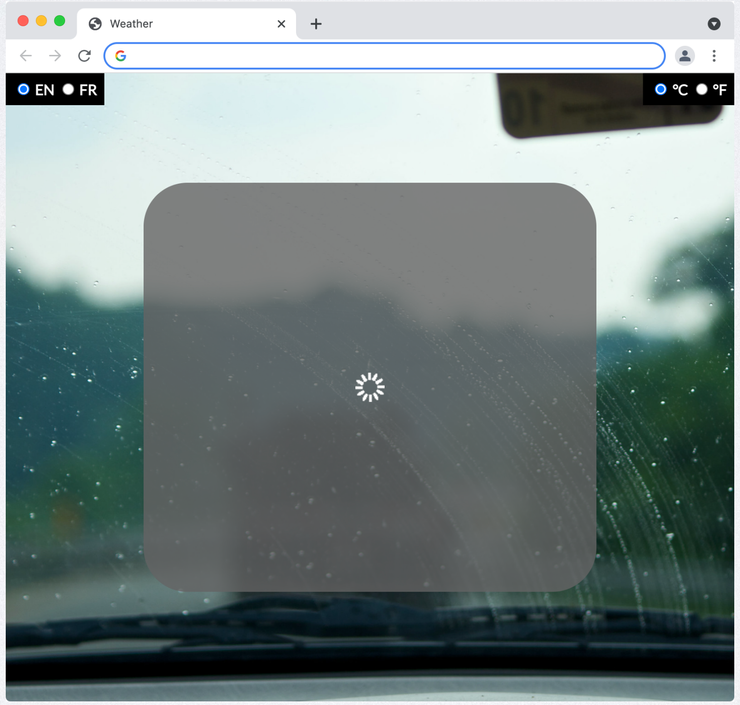
Part 4

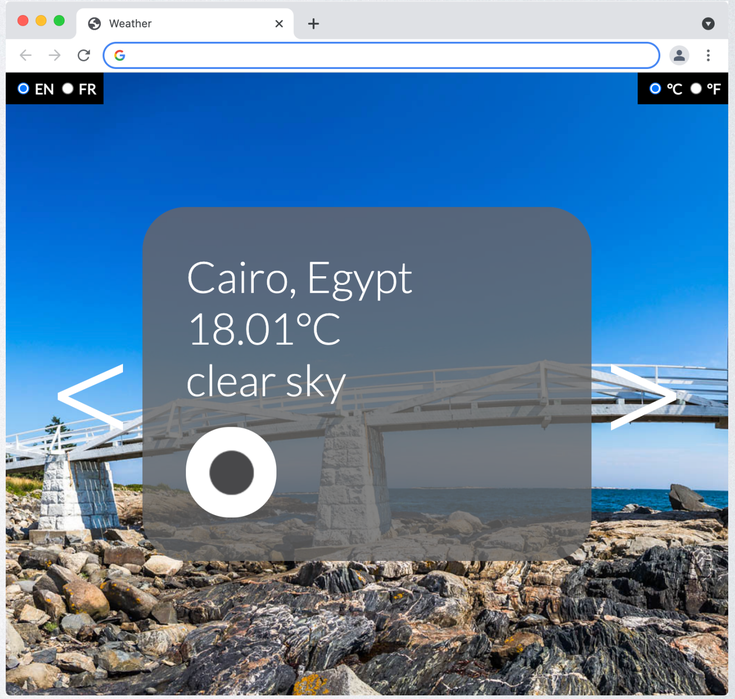
When you are getting new details from the web service...

... you should indicate you are waiting on the service with a loader gif replacing the previous content...

... which should be replaced by the new content once it has arrived.







See screencasts to see it all in action.

Technical Tips

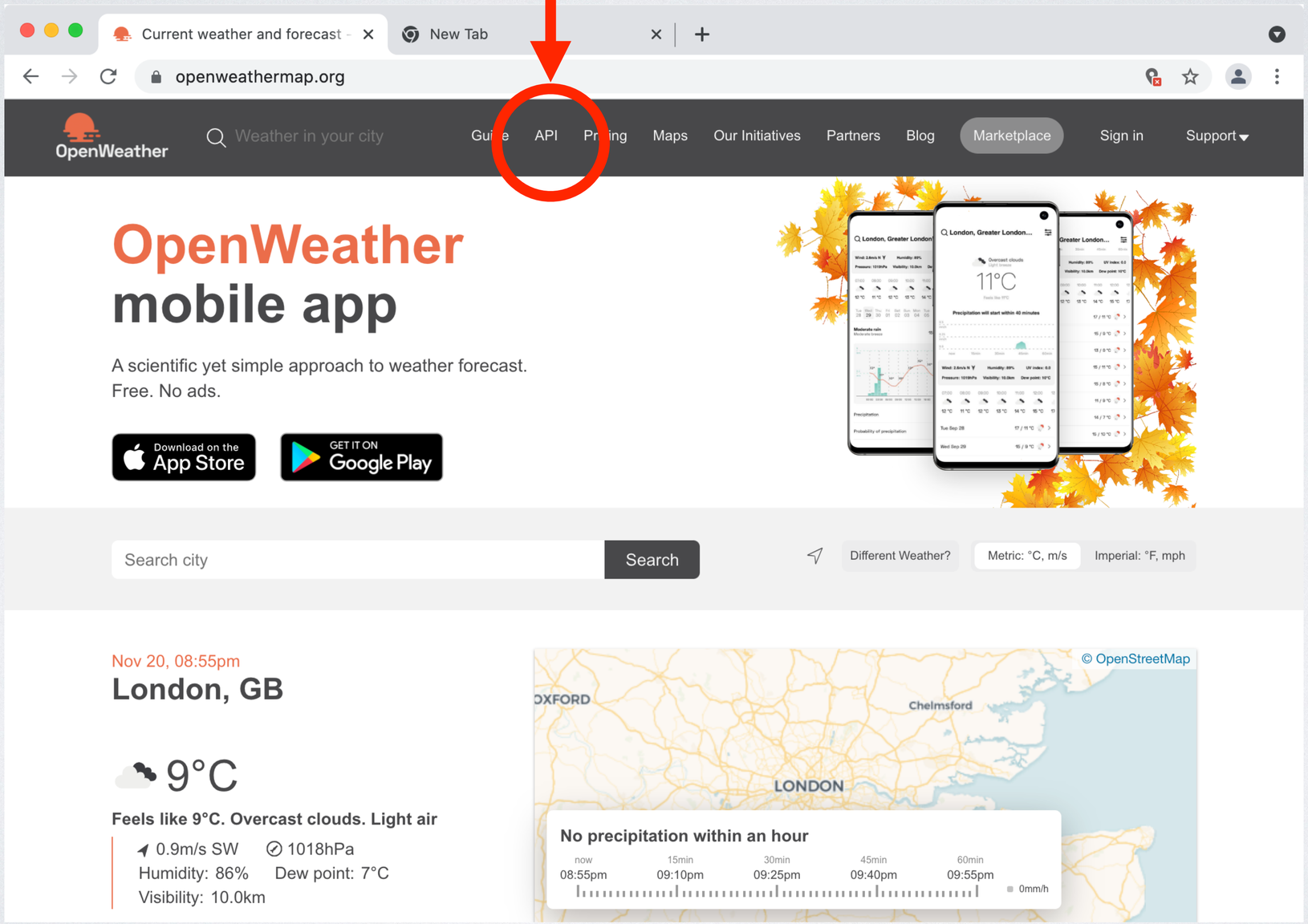
API/ Web Service

We will be using the **openweathermap.org** API.

You will need an API key to use the service.

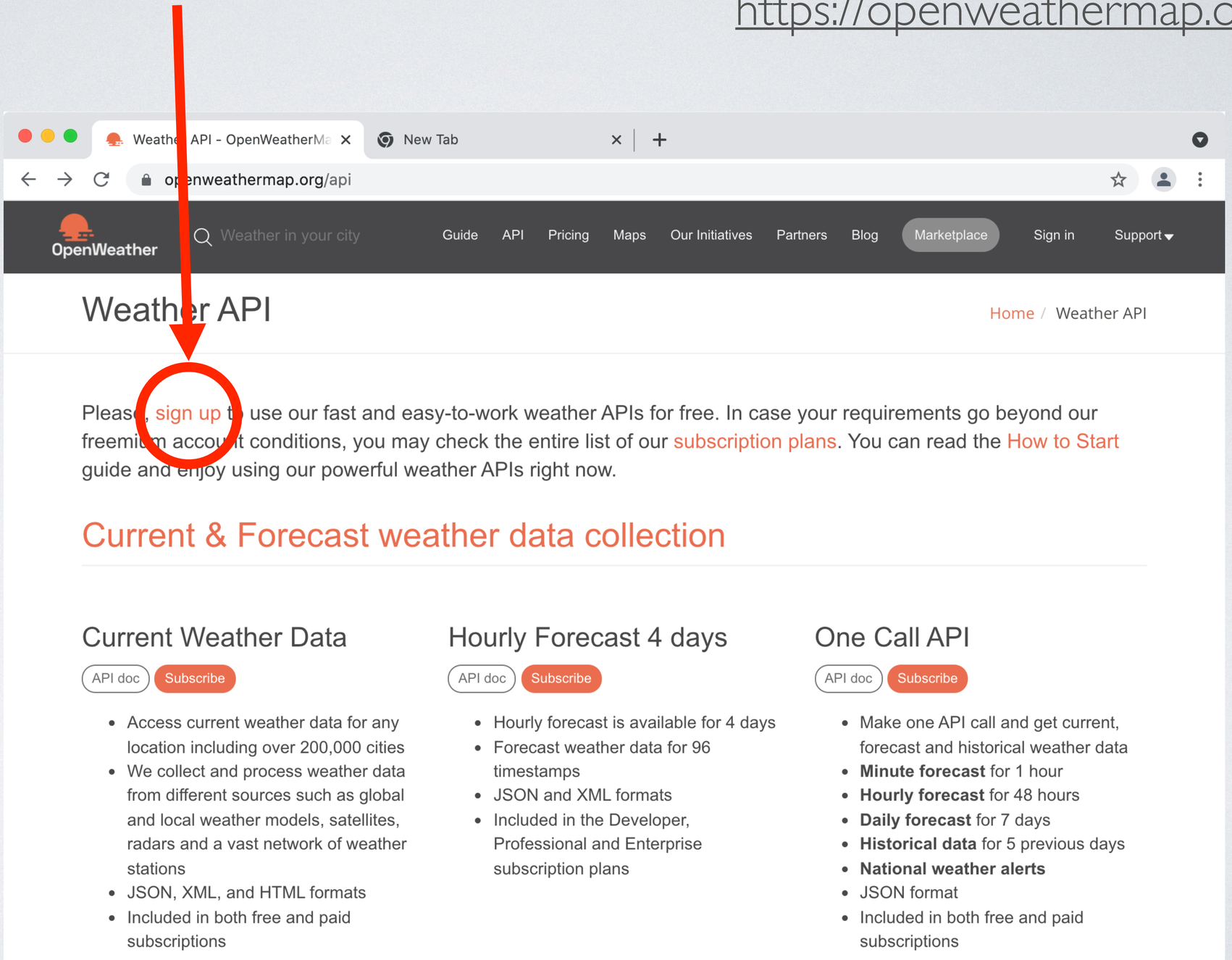
Go to https://openweathermap.org

Click on the **API** link

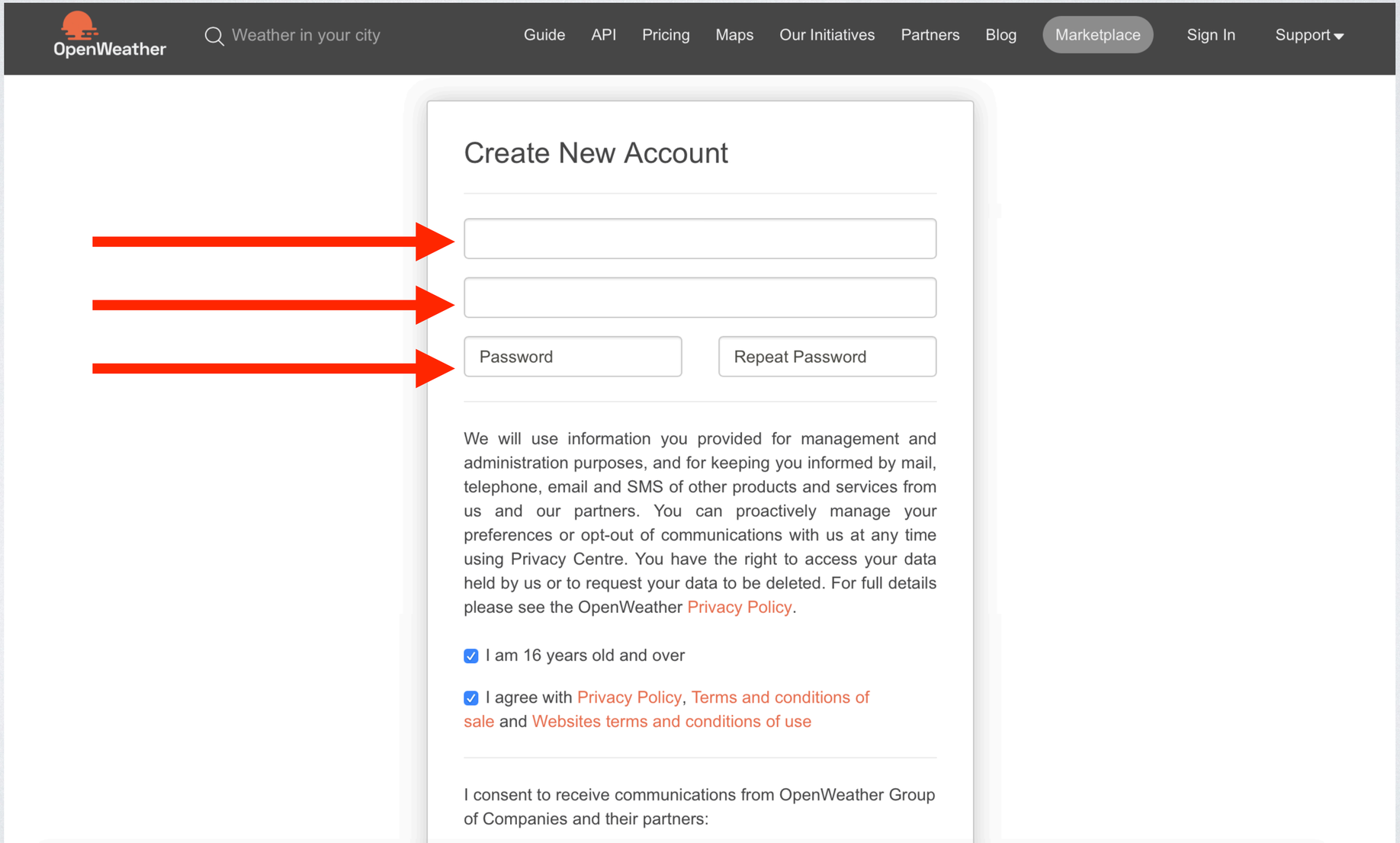


Click on the "sign up" link to create a key.

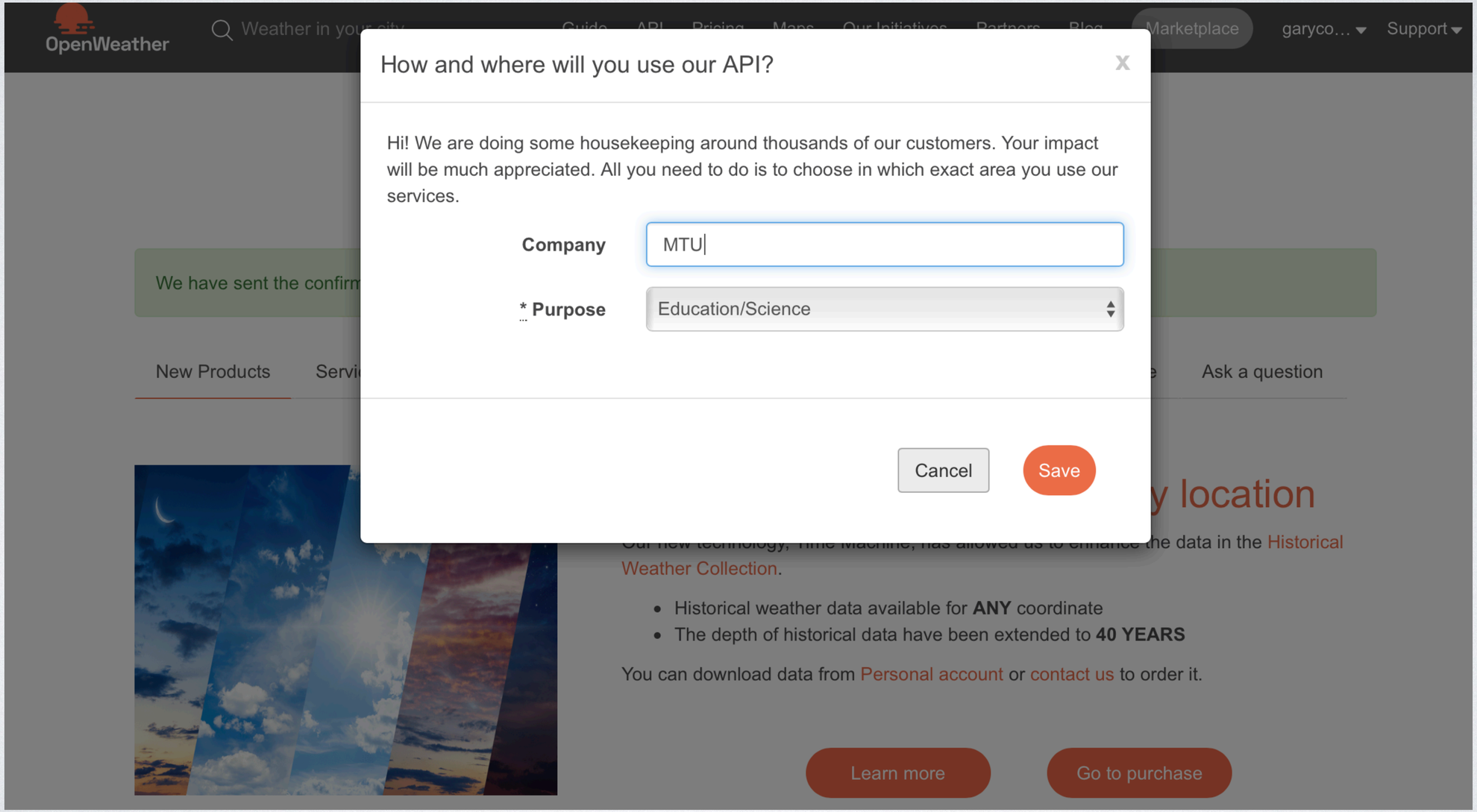
https://openweathermap.org/api



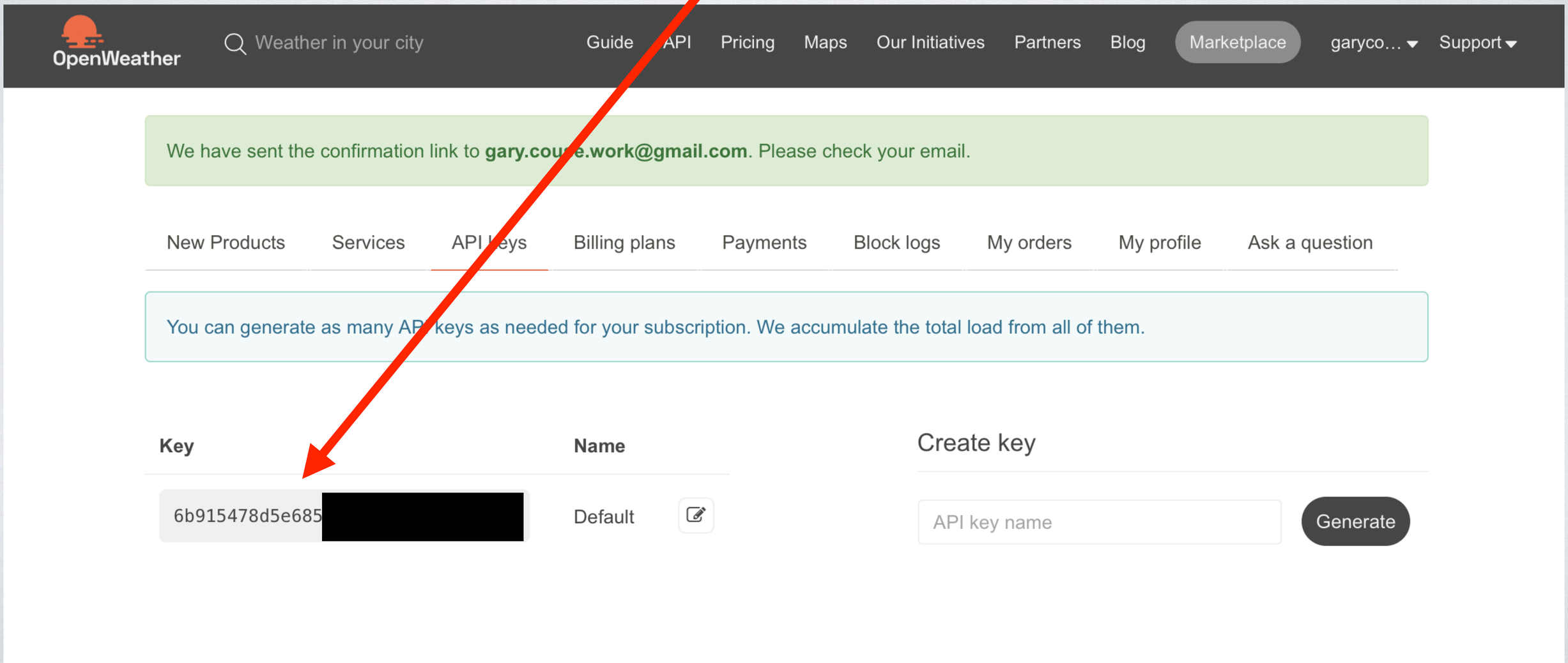
Enter your own details (name + email + password + repeat password)



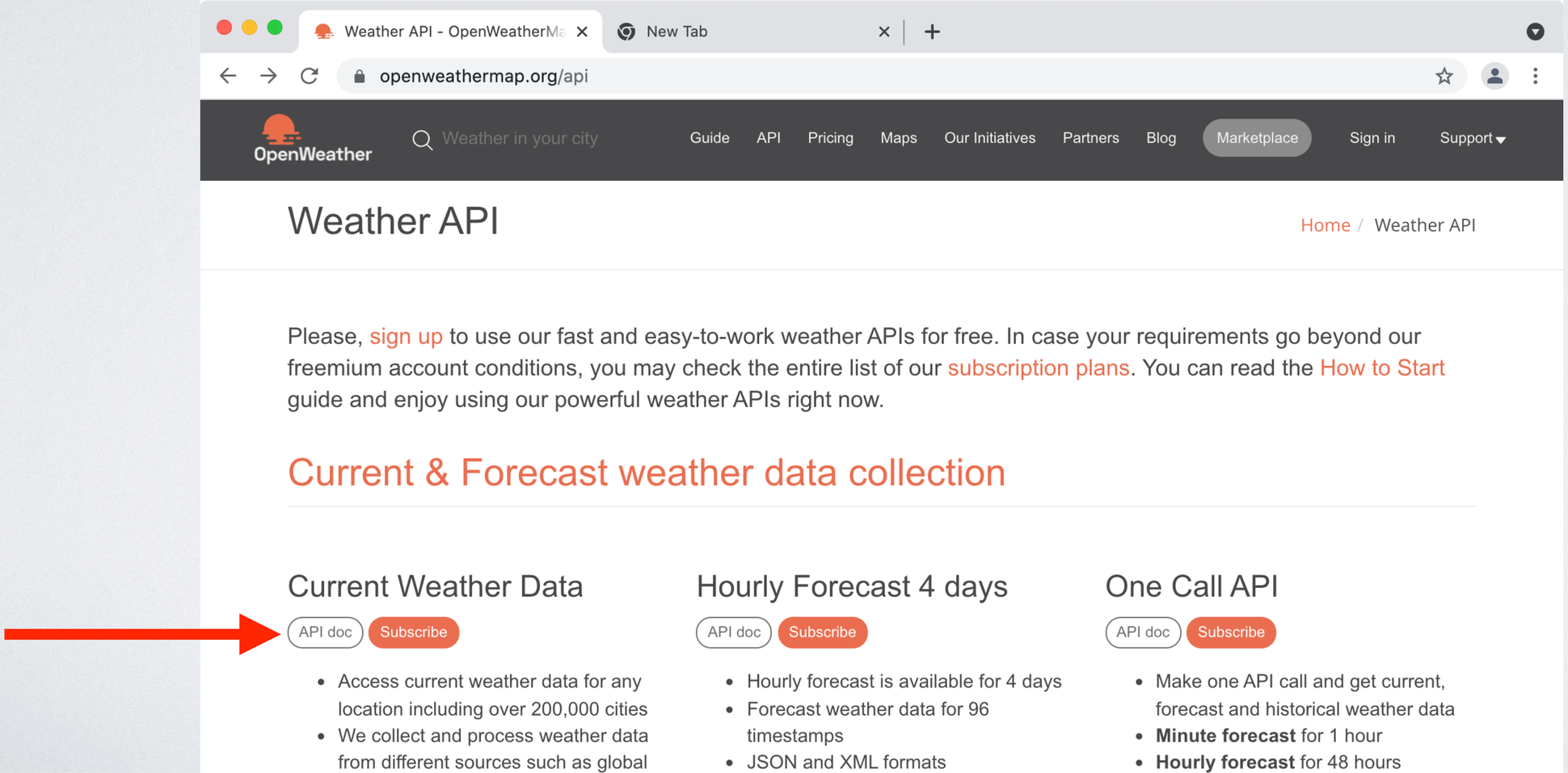
You will be asked to describe what you want to use the API for.



You will then see the dashboard with your API key. (partially redacted here)



Go back to the **API** page and click on the "API doc" link for the **Current Weather Data** option.



This page describes the data you send to the service, and the data you get back.

Current weather data - OpenW | New Tab

openweathermap.org/current

OpenWeather | Weather in your city | Guide | API | Pricing | Maps | Our Initiatives | Partners | Blog | Marketplace | Sign in | Support

Current weather data

Home / API / Current weather

Access current weather data for any location on Earth including over 200,000 cities! We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations. Data is available in JSON, XML, or HTML format.

Call current weather data for one location

By city name

You can call by city name or city name, state code and country code. Please note that searching by states available only for the USA locations.

API call

```
api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
```

```
api.openweathermap.org/data/2.5/weather?q={city name},{state code}&appid={API key}
```


Call current weather data for one location
- By city name
- By city ID
- By geographic coordinates
- By ZIP code

Call current weather data for several cities
- Cities within a rectangle zone
- Cities in circle

Bulk downloading

Weather fields in API response
- JSON
- XML
- List of condition codes
- Min/max temperature in current weather API and forecast API

Other features
- Format
- Units of measurement
- Multilingual support
- Call back function for JavaScript code

URLS (API Documentation).

Specifying the City you want the weather for.

https://openweathermap.org/current#name

Specifying Centigrade or Fahrenheit

https://openweathermap.org/current#data

Specifying language for the description

https://openweathermap.org/current#multi

You pass the relevant information to the API in the request URL.

http://api.openweathermap.org/data/2.5/weather?callback=**process**&lang=**fr**&units=**metric**&q=**Cork,Ireland**&appid=**6b93217**

Base URL

`http://api.openweathermap.org/data/2.5/weather?`

You pass the relevant information to the API in the request URL.

metric or **imperial**

en or **fr**

Name of the function you want called when the information comes back.

Search term

Your own api key

URLS (API Documentation).

The response format.

https://openweathermap.org/current#current_JSON

Information about icons and weather codes.

https://openweathermap.org/weather-conditions

The data comes back like this:


```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
```

We want the following bits of information.


```
{
  "coord": {
    "lon": -122.08,
    "lat": 37.39
  },
  "weather": [
    {
      "id": 800,
      "main": "Clear",
      "description": "clear sky",
      "icon": "01d"
    }
  ],
  "base": "stations",
  "main": {
    "temp": 282.55,
    "feels_like": 281.86,
```


Weather code (for background image)

Weather Description

Icon Code

Temperature

If the response to your query is in a variable called **data** then the data we want is accessible as follows:

```
var weatherCode = data.weather[0].id;
var weatherDescription = data.weather[0].description;
var weatherIcon = data.weather[0].icon;
var weatherTemperature = data.main.temp;
```

https://openweathermap.org/weather-conditions

Weather Codes

Weather codes in the 200s are all considered Thunderstorms. Codes in the 300s are considered Drizzle, etc. Below are the 7 main categories of weather. You should have a background image corresponding to each of these.

Group 2xx: **Thunderstorm**
Group 3xx: **Drizzle**
Group 5xx: **Rain**
Group 6xx: **Snow**
Group 7xx: **Atmosphere**
Group 80x: **Clouds**
(*Note the exception here that* weather code 800 is considered **Clear**)

https://openweathermap.org/weather-conditions

Icons

To display an icon you get the icon code from the response.

E.g. 10d

You then use this code to construct a URL you can use in an image tag or CSS.

https://openweathermap.org/img/wn/10d@2x.png

On dark background

icon code

https://openweathermap.org/img/wn/10d@2x.png

On dark background

Coding

An easy way to code this application is to write a function that sends a request to **openweathermap** (it would construct the request URL, and send it using the JSON-P technique). In order to construct the URL it gathers the information it needs from various variables/form elements.

If you make any changes to what you want displayed (*next button*, *previous button*, *language selection*, t*emperature scale selection*) just call that function and it will create an updated request URL. When the response comes back it will refresh the data on the page.

CSS

Depending on your design (HTML, CSS and/or JavaScript) when you replace the data on screen for the loading gif you may find the display area shrink if the original content has been removed (e.g. with the **display** CSS property set to **none**). An alternative might be the **visibility** property. Like **display** it can make an element disappear. Unlike with **display**, the element is still considered to be on the page and takes up space. It just can't be seen.

E.g.

visibility: visible;

or

visibility: hidden;

Preloading

Since the background images are large it may be sensible to preload them before you need them (to avoid a delay when showing them first).

You can do this in JavaScript when the page first loads. Setting the **src** property forces the image to be downloaded.

```
var thunderstorm = new Image();
thunderstorm.src = "thunderstorm.jpg";

var drizzle = new Image();
drizzle.src = "drizzle.jpg";
```

etc. etc.

Whenever you are using the image filename you should use the **src** property of the Image project. This means you will be using the preloaded version.

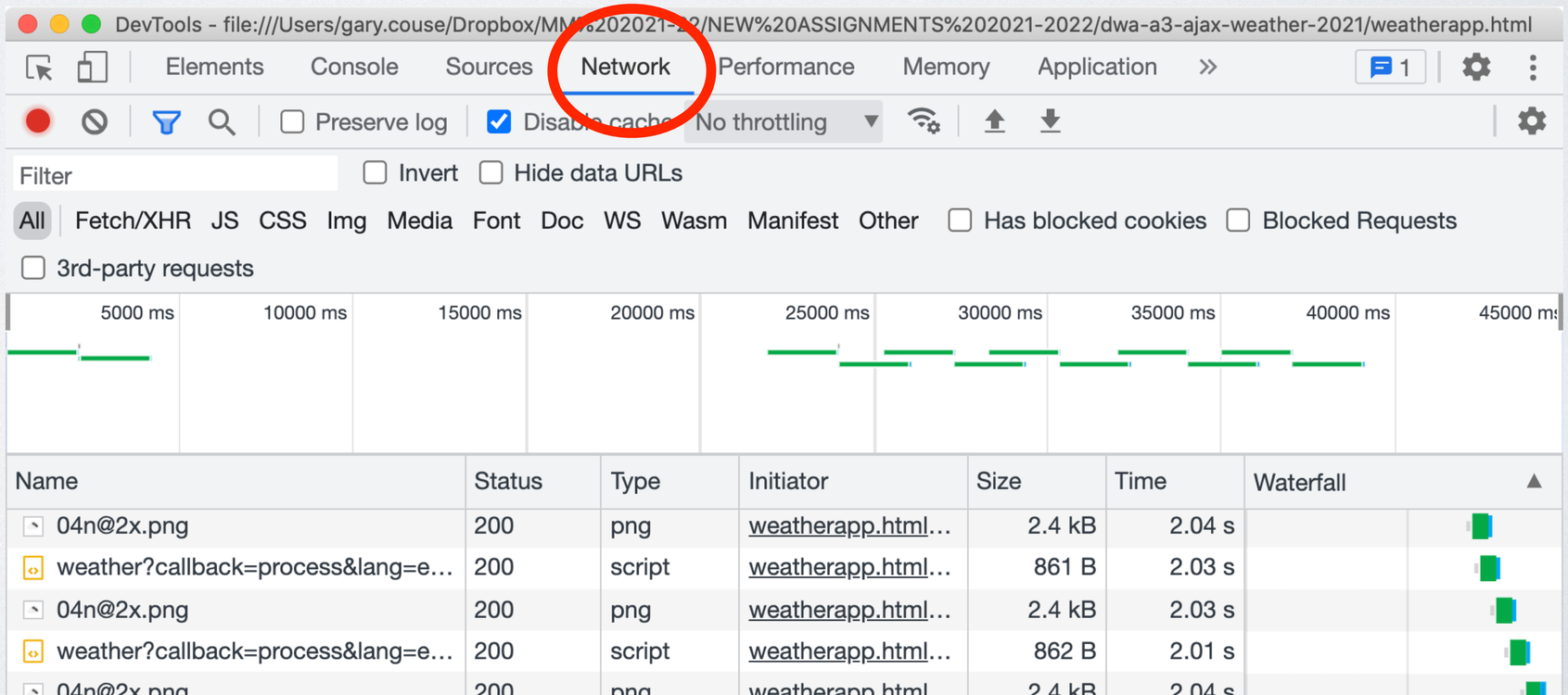
```
document.getElementById("con").style.backgroundImage = "url(" + thunderstorm.src + ")";
```

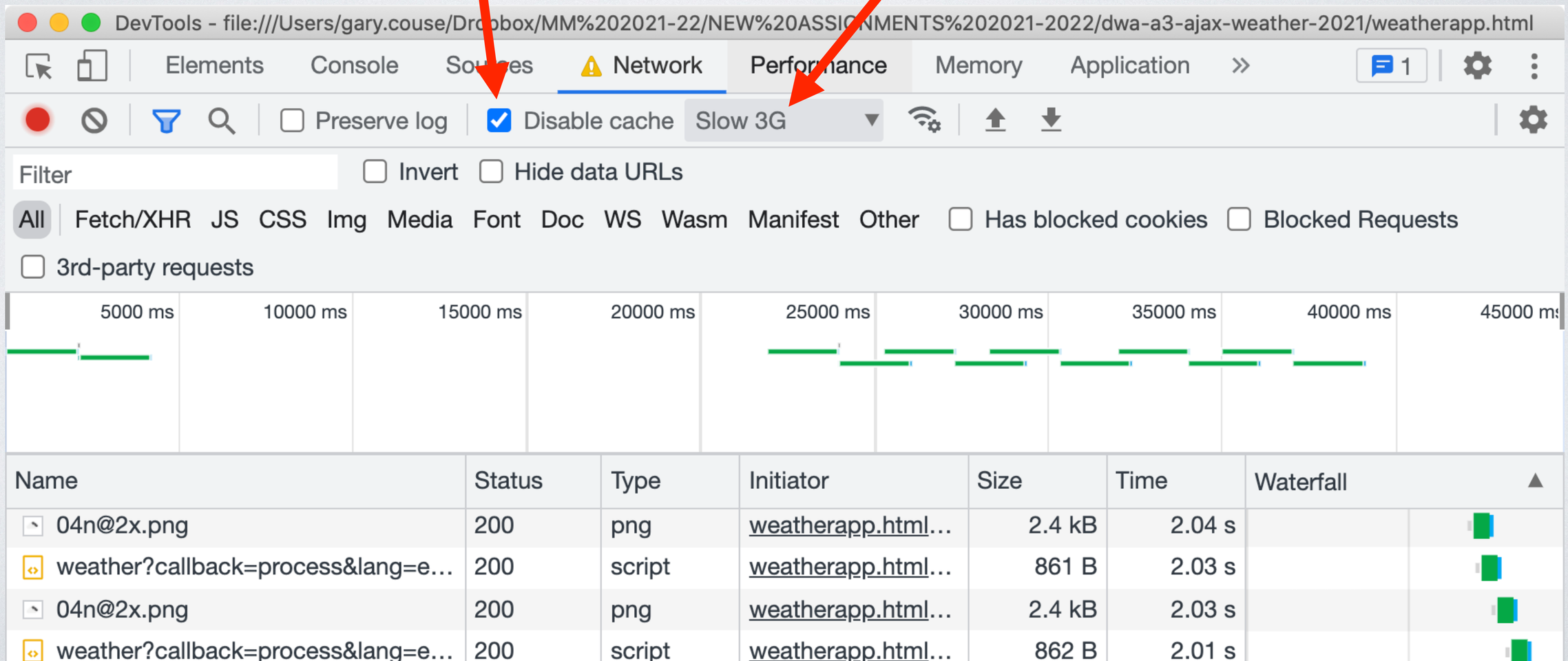
Web Development Tools

It may help when showing the loader gif to slow down the browser so you can see it before the data comes back from **openweathermap.**

You can have the browser pretend it has a slow internet connection by opening up the developer tools (e.g. right click on the page and choose **inspect**).

Click on **Network**



Choose "Slow 3G" from this menu.

Disable Cache