|  |  |
| --- | --- |
| $(document).ready(**function** ()   $("#regions").change(**function** () {  **let** region = $(**this**).val();  **switch** (region) {  **case** 'england':  $("#cities").load("england-cities.html");  **break**;  **case** 'scotland':  $("#cities").load("scotland-cities.html");  **break**;  **case** 'wales':  $("#cities").load("wales-cities.html");  **break**;  **case** 'nirland':  $("#cities").load("nireland-cities.html");  **break**;  }    $("#cities").change(**function** () {  **let** city = $(**this**).val();    **let** apiUrl = `https:\/\/api.openweathermap.org/data/2.5/weather?q= ${city},uk&APPID=414c5f64311ce1d3548d02e08f7fe1a6`;  $.ajax({  url: apiUrl,  type: "GET",  dataType: "json",  success: **function** (response) {  **let** lastUpdated = **new** Date(response.dt \* 1000);  **let** dd = lastUpdated.getDate();  **let** mm = lastUpdated.getMonth() + 1;  **let** yyyy = lastUpdated.getFullYear();  **if** (dd < 10) {  dd = '0' + dd  }  **if** (mm < 10) {  mm = '0' + mm  }  **let** date = dd + '-' + mm + '-' + yyyy;   **var** sTxt = `<dl id='list'><dt>Name:</dt><dd> ${response.name} </dd><dt>Date:</dt><dd> ${date} </dd><dt>Weather condition:</dt><dd>${response.weather[0].description}</dd>`;      **let** img = **new** Image();  img.src = "http://openweathermap.org/img/wn/" + response.weather[0].icon + "@2x.png";  sTxt += `<dd> ${img.outerHTML} </dd><dt>Temperature:</dt>`;  **let** metric = (response.main.temp -273.15).toFixed(1);  **let** imperial = (1.8 \* metric + 32).toFixed(1);  **if** ((metric > 35) || (metric < -5)) {  sTxt += "<dt id='warn'>Warning: Severe weather conditions</dt>";  }  sTxt += `<dd> ${metric}°C</dd><dd> ${imperial}F</dd><dt>Wind speed:</dt>`;    **let** kph = ((response.wind.speed \* Math.pow(60, 2)) / 1000).toFixed(1);  **if** (kph > 50) {  sTxt += "<dt id='warn'>Warning: Severe winds</dt>";  }    sTxt += "<dd>" + kph + " kph</dd><dd>" +  ((response.wind.speed \* 0.000621371) \* Math.pow(60, 2)).toFixed(1) + " mph</dd><dt>" + "Wind direction:</dt><dd>" + response.wind.deg + "°</dd>";    apiUrl = `https:\/\/api.openweathermap.org/data/2.5/weather?q= ${city} ,uk&mode=xml&APPID=414c5f64311ce1d3548d02e08f7fe1a6`;  $.ajax({  url: apiUrl,  type: "GET",  **async**: **false**,  dataType: "xml",  success: **function** (response) {  **let** direction = $(response).find('direction').attr('name');  sTxt += "<dd>" + direction + "</dd>";  },   error: **function** (xhr, error) {  $("#info").append(error.toUpperCase() + ". HTTP status request to get wind direction failed" + xhr.status);  sTxt += "<dd>undefined</dd>";  }  });    sTxt += "</dl>";    $("#cityInfo").html("");    $("#cityInfo").append(sTxt);  **if** ($("#cityInfo").attr("hidden")) {  $("#cityInfo").show();  }  },  error: **function** (xhr, error) {  $("#info").append(error.toUpperCase() + ". HTTP status" + xhr.status +". Failed to fetch data");  }  });  });  }); }); | //a function that runs after the page is loaded.  //we attach an anonymous function that listens and runs when a change is detected on the DOM node containing the id ‘regions’.  //The value that the user selected is assigned to the variable *region*.  //Then, based upon what the value (i.e. country within the UK) of ‘region’ is, the html file that contains the list of cities held for this value is loaded into the node associated with the id ‘cities’.  //similarly, we now attach a function that will detect and pick up the value that the user selected from the second drop-down. The second drop down lists the cities for each region.  //we construct the url to the API we would like to query, concatenating to it, the relevant city and API code.  // a jQuery function to issue an (asynchronous) http request to get data, in json format, from a source (the api link we just constructed). It takes an object as a parameter with five properties: the url link, the type of request – GET. the expected data format for the data returned and two callback functions. One that will run in case of a successful request and return and within that function we will define exactly what happens.  Alternatively, In case of an error, the second function will output a string describing the error and identify the status code of the [jqXHR](http://api.jquery.com/Types/#jqXHR) object.  //extracts the time of date of the json weather report (as detailed here <https://openweathermap.org/current)>. This is given in UNIX format and we multiply it by 1000 in order to convert it to a Date object that takes milliseconds. This allows for specific extraction of the date.  //0, is for January.  //in case of single digit days.  //in case of single digit months.  //we build the sTxt variable (a function-scoped string variable) to accumulate the html. This is needed later for insertion into the html so that the results can be displayed to the user. Here, we build a data definition list. This list comprises the dt tags holding the headers or the names and the dd tags containing the values (i.e. the various values obtained from the ‘response’ object). JavaScript’s template strings and interpolation are utilized to build the html string and to insert the variable’s into the middle of it, wherever appropriate. We start off with extracting city name, date, and weather description. The last one obtained from a property called *description* that is found within the first index of an array called *weather*, contained within the *response* object.  //the Image object is assembled, and we fill its src attribute. The img will point to the icon via link. The value of the particular icon, we get from the json data acquired (as detailed here <https://openweathermap.org/weather-conditions>).  //The image tag is glued to xTxt.  //as the temperature is defined in kalvin it is initially converted into metric and then from metric to imperial rounding both numbers to 1 decimal point.  //we check if the temperature is within the bounds. If it is found to be outside of the parameters a warning message is raised.  /the wind speed is measured in meters per second and requires conversion to meters per hour and then from meters to mph so that mph can also be shown.  //again, we test to check if we should generate a warning for severe weather conditions. i.e. strong winds.  // kph is converted to mph.  // information concerning the wind direction in degrees is obtained and inserted.  //at this point an ajax request is made, this time an **xml** request. This is to obtain and display the code for the wind direction in string (eg. South etc) as the json object does not provide that information as is, and there could be a possible 16 values for it.  //the function is set to be asynchronous in order to ensure the sTxt variable is available and workable inside this ajax function.  //we find the ‘direction’ node in the received xml object and we obtain the value of its attribute called ‘name’ as detailed here. <https://openweathermap.org/current>.  //the definition of the error function inside the nested ajax function determines what to do in case of an error happening to our request. (As we have most of the data, we want the other data still to be shown so we go ahead and finish off the data definition list, except we also display a specific error msg, re the wind direction data fetch failure, to the user.)  //closing the data definition list.  //clearing any previously held data from the html node associated with the id ‘cityInfo’ - that is the node (div) - that will update the user about the newly refreshed weather data.  //we append our html-string results to that div.  //if its hidden attribute was set, like when before the user first selects a viewing, the function then unhides it.  //the definition of the error function, in the outer ajax function. |