Front-End Essentials

Lab Guides

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.1**
Effective Date	**20/11/2012**

Hanoi, 04/2019

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	25/Jun/2018	Create a new Lab	Create new	DieuNT1	VinhNV
2	01/May/2019	Update Fsoft Template	Update	DieuNT1	VinhNV

Contents

CODE:	FEE.M.L402 (Weather App)
TYPE:	Medium
LOC:	150
DURATION:	90 MINUTES

Unit 4 – jQuery and AJAX

Objectives

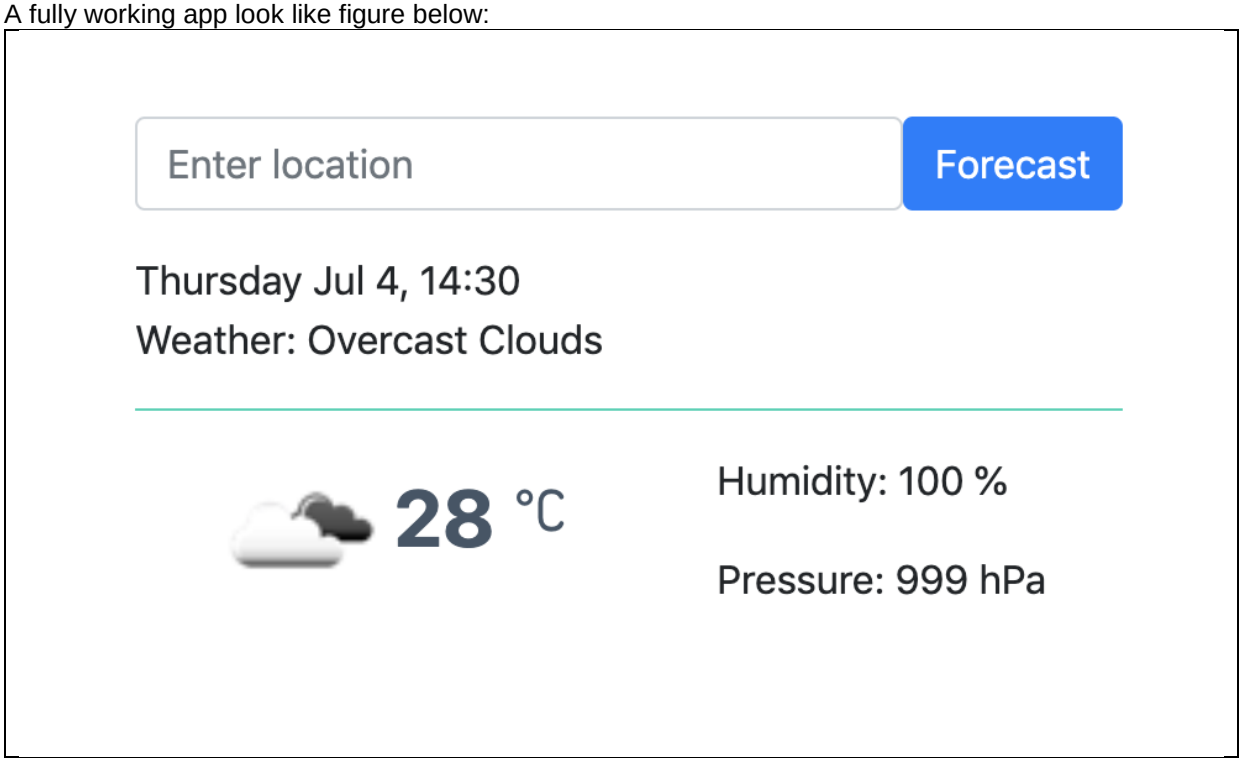
- Understand the core concepts of JavaScript programming language
- Understand basic concept of DOM
- Able to add behavior to make web site dynamic using JavaScript (DOM)
- Understand the core concepts of Bootstrap (layout, rows, grid, flex, components: buttons, alerts, utilities)
- Understand jQuery (selector, onclick, add/remove attribute, toggle, insert, remove class, GET, POST) and AJAX
- Able to use jQuery and AJAX to interactive with Web API

Technical Requirements:

- Must use HTML, CSS, and Bootstrap 4
- Must use jQuery to interact with Web API

Specifications

You have to build a Weather App using JavaScript and jQuery. The app will display the weather of user by locating the user automatically or user can input location (city name) to forecast weather.
A fully working app look like figure below:

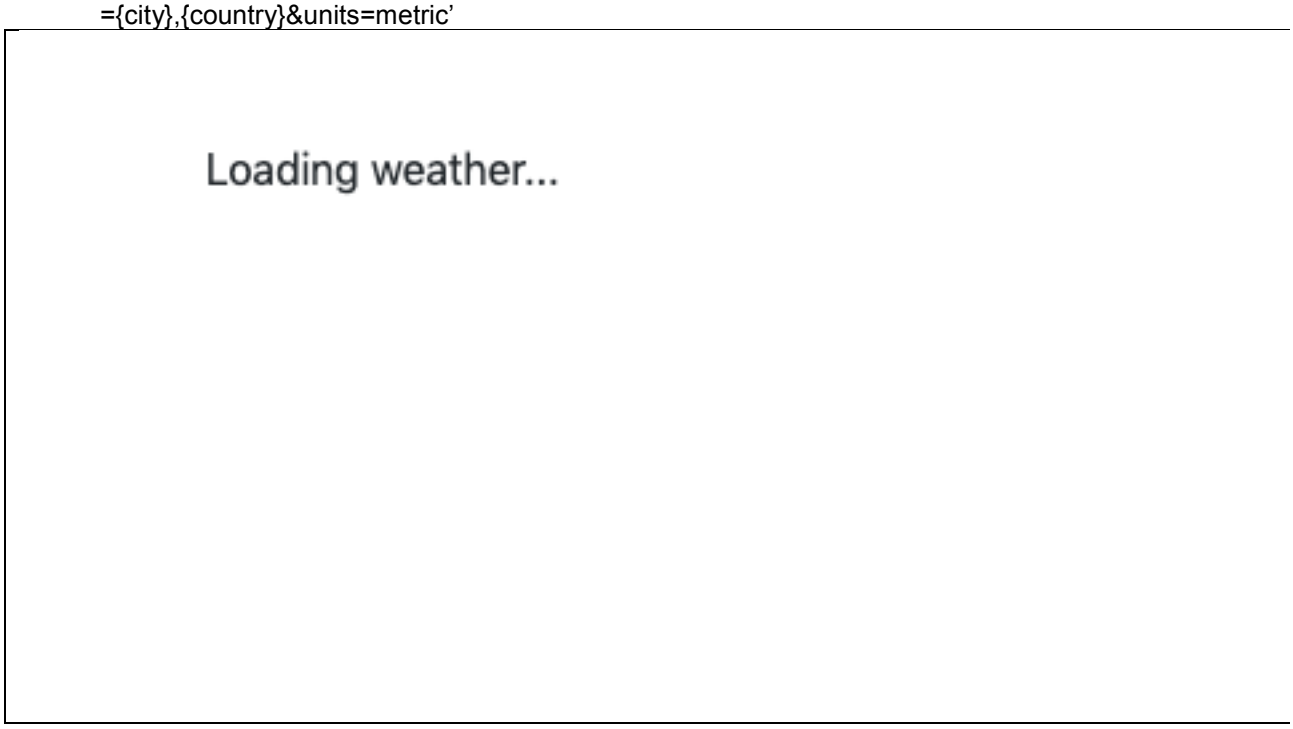


The app will work as below:

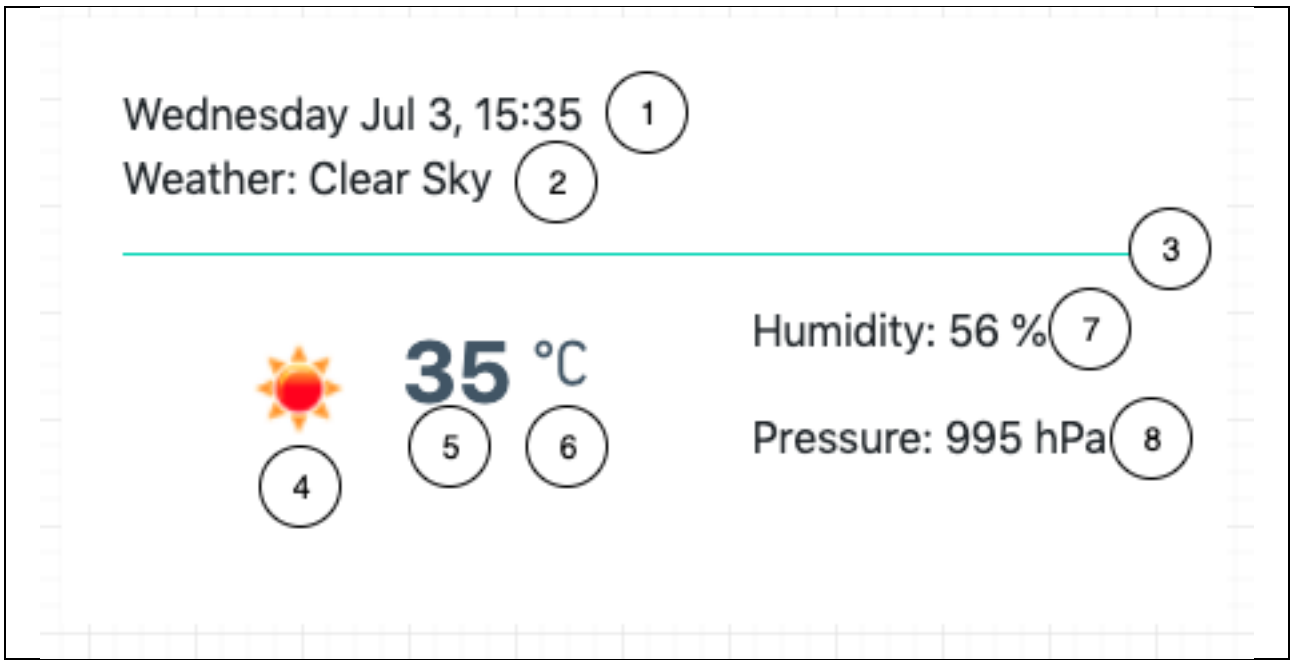
- After HTML is loaded, show in a text “Loading location…” Meanwhile, you have to query for current location using this Location API ‘http://ip-api.com/json’ using GET HTTP Method and extract the value ‘city’ and ‘country’ from the Location API response to use later

- After retrieve the Location API response, you must hide the text “Loading location…” and show a new text “Loading weather…”. This time, using the **city** and **country** from previous step, you must query for weather data from this Weather API

  ‘https://api.openweathermap.org/data/2.5/weather?appid=6c186bd312fb6c44839158e1da4c8d1e&q={city},{country}&units=metric’



- After retrieve the response from weather API, you must show the weather like figure below (every data is presented in the response from weather API)

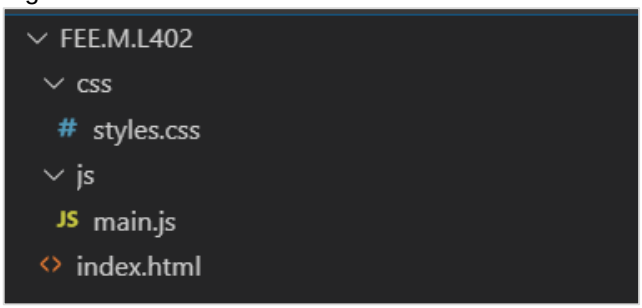


- Item 1 (Wednesday Jul 3, 15:35) is the formatted string of current date
- Item 2 (Weather: Clear Sky): the description of weather retrieved from Weather API response
- Item 3: horizontal rule
- Item 4: the icon matching current weather. Check value icon from Weather API response and the image is from 'http://openweathermap.org/img/w/{icon}.png'
- Item 5: the temperature
- Item 6: Icon indicates unit for temperature (Celsius)
- Item 7: Humidity in percentage
- Item 8: Pressure in hPa

Guidelines

Step 1: Create project structure

Create project structure like figure below:



Step 2: Open project in IDE

Open newly created project in **Visual Studio Code**

Step 3: Create index page

Open file **index.html** in VSC (Visual Studio Code), and type in html VSC will show a list of suggestions. Choose **html:5** and press Enter.
Change value of **title** tag to **Weather App**

Step 4: Add Bootstrap 4 and Custom CSS

Add Bootstrap 4 CDN link, Weather Icons and **styles.css** file into head section of **index.html** file

```
1. <!DOCTYPE html>
2. <html>
3. <head>
4.   <meta charset="utf-8">
5.   <title>Weather App</title>
6.   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
7.
8.   <link rel="stylesheet"
   href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css"
   integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
9.     crossorigin="anonymous">
10.  <link rel="stylesheet" type="text/css"
11.    href="https://cdnjs.cloudflare.com/ajax/libs/weather-icons/2.0.9/css/weather-
   icons.min.css" />
12.  <link rel="stylesheet" href="css/styles.css" media="screen" charset="utf-8">
13. </head>
```

Step 5: Add JavaScript

Add jQuery and **main.js** to **index.html** file by append a **script** tag before closing tag of **body**

```
1. <body>
2.   <script src="https://code.jquery.com/jquery-3.4.1.min.js"
3.     integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
4.     crossorigin="anonymous"></script>
5.   <script src="js/main.js"></script>
6. </body>
```

Step 6: Create HTML layout

Our next step is to implement the page layout like design, we use of Bootstrap CSS to make it easier to create the layout:

```
1. <body>
2.   <div id="weather-app">
3.     <form id="form">
4.       <div class="form-group d-flex justify-content-between">
5.         <input type="text" class="form-control" id="form-input"
6.           aria-describedby="emailHelp" placeholder="Enter location">
7.         <button class="btn btn-primary" type="submit">Forecast</button>
8.       </div>
9.     </form>
10.    <div class="well no-select">
11.      <div id="loading-location">Loading location...</div>
12.      <div id="loading-weather">Loading weather...</div>
13.      <div class="current-date" id="current-date"></div>
14.      <div class="weather-detail" id="weather-detail"></div>
15.    </div>
16.  </div>
17.  <script src="https://code.jquery.com/jquery-3.4.1.min.js"
18.    integrity="sha256-CSXorXvZcTkaix6Yvo6HppcZGetbYMGWSFlBw8HfCJo="
19.    crossorigin="anonymous"></script>
20.  <script src="js/main.js"></script>
21. </body>
```

Open **index.html** in Live Server to take a look.

Step 7: Declare DOM Objects

Next step, we declare DOM Objects that help us to handle user input:

```
1. $(document).ready(function() {
2.   // Declare DOM Object
3.   var form = $('#form');
4.   var input = $('#form-input');
```

```
5.    var $currentDate = $('#current-date');
6.  });
7.
```

Step 8: Bind Events

Since the input text and Submit button are in a form, whenever user press Enter or click on Submit a submit, an submit event will be emitted, the code below handle such event:

```
1.  $(document).ready(function() {
2.    // Declare DOM Object
3.    var form = $('#form');
4.    var input = $('#form-input');
5.    var $currentDate = $('#current-date');
6.
7.    // Handle user input
8.    form.on('submit', function(event) {
9.      event.preventDefault();
10.
11.     console.log('test', input.val());
12.   });
13. });
```

Step 9: Get Location

To get the location of user, we must call to Web API using $.ajax
$.ajax return a Promise containing the response from Web API:

```
1.  function paddingLeft(n) {
2.      return n < 10 ? `0${n}` : n;
3.    }
4.
5.    function getCurrentDate() {
6.      const days = [
7.        'Sunday',
8.        'Monday',
9.        'Tuesday',
10.       'Wednesday',
11.       'Thursday',
12.       'Friday',
13.       'Saturday'
14.     ];
15.     const months = [
16.       'Jan',
17.       'Feb',
18.       'Mar',
19.       'Apr',
20.       'May',
21.       'Jun',
22.       'Jul',
23.       'Aug',
24.       'Sep',
25.       'Oct',
26.       'Nov',
27.       'Dec'
28.     ];
29.     const now = new Date();
30.
31.     return `${
32.       days[now.getDay()]
33.     } ${months[now.getMonth()]} ${now.getDate()},
    ${paddingLeft(now.getHours())}:${paddingLeft(now.getMinutes())}`;
34.   }
35.
36.   function ipLookUp() {
37.     $.ajax('http://ip-api.com/json').then(function success(response) {
```

```
      $('#loading-location').hide();
      $('#loading-weather').show();

      console.log(response);

      $currentDate.html(getCurrentDate());
    });
  }
```

In the code above, we create 2 helper function that help us to format the current date.
With that data, we simply display it to HTML using jquey.html() method

Step 10: Get Weather Information

Try to call ipLookUp function and check the console for response from Web API.
You should see something similar:

```
▼ Object ⓘ
    as: "AS18403 The Corporation for Financing & Promoting Technology"
    city: "Da Nang"
    country: "Vietnam"
    countryCode: "VN"
    isp: "FPT-STATICIP"
    lat: 16.0544
    lon: 108.202
    org: "FPT Telecom Company"
    query: "42.116.166.25"
    region: "27"
    regionName: "Quảng Nam"
    status: "success"
    timezone: "Asia/Ho_Chi_Minh"
    zip: ""
  ▶ __proto__: Object
```

To query for Weather, we will need to extract the city and country value from response and call to Weather API to get the weather forecast:

```
  function fetchWeather({
    city = 'Hanoi',
    country = 'Vietnam',
    units = 'metric'
  }) {
    return $.ajax(`${WEATHER_API_URL}&q=${city},${country}&units=${units}`);
  }
```

Step 11: Render Weather Information

Same like when you get IP Information, once you get response from Weather API, you need to render it to the HTML

```
function renderWeather(weather) {
    const { icon, description } = weather.weather[0];
    const { temp, humidity, pressure } = weather.main;
    const { name } = weather;

    $('#weather-detail').html(`
        <div>City: ${name}</div>
```

```
8.          <p class="text-capitalize">Weather: ${description}</p>
9.          <hr/>
10.         <div class="weather-detail-container">
11.           <div class="item toggle-units">
12.           <img class="img-lg" src="http://openweathermap.org/img/w/${icon}.png" />
13.           <span class="text-lg">${temp}</span>
14.           <div class="icon">
15.             <i class="wi wi-celsius icon-lg"/>
16.           </div>
17.         </div>
18.
19.         <div class="item">
20.           <p>Humidity: ${humidity} %</p>
21.           <p>Pressure: ${pressure} hPa</p>
22.         </div>`);
23.   }
```

Step 12: Search for Weather Information from User input

Check the code below, as we handle user input and feed it to fetchWeather

```
1.  $(document).ready(function() {
2.    const WEATHER_API_KEY = '6c186bd312fb6c44839158e1da4c8d1e';
3.    const WEATHER_API_URL =
    `https://api.openweathermap.org/data/2.5/weather?appid=${WEATHER_API_KEY}`;
4.
5.    function paddingLeft(n) {
6.      return n < 10 ? `0${n}` : n;
7.    }
8.
9.    function getCurrentDate() {
10.     const days = [
11.       'Sunday',
12.       'Monday',
13.       'Tuesday',
14.       'Wednesday',
15.       'Thursday',
16.       'Friday',
17.       'Saturday'
18.     ];
19.     const months = [
20.       'Jan',
21.       'Feb',
22.       'Mar',
23.       'Apr',
24.       'May',
25.       'Jun',
26.       'Jul',
27.       'Aug',
28.       'Sep',
29.       'Oct',
30.       'Nov',
31.       'Dec'
32.     ];
33.     const now = new Date();
34.
35.     return `${
36.       days[now.getDay()]
37.     } ${months[now.getMonth()]} ${now.getDate()},
    ${paddingLeft(now.getHours())}:${paddingLeft(now.getMinutes())}`;
38.   }
39.
40.   function ipLookUp() {
41.     $.ajax('http://ip-api.com/json').then(function success(response) {
42.       $('#loading-location').hide();
43.       $('#loading-weather').show();
44.
45.       console.log(response);
```

```
46.       const { city, country } = response;
47.
48.       $currentDate.html(getCurrentDate());
49.
50.       fetchWeather({ city: city.replace(/\s+/g, ''), country }).then(renderWeather);
51.     });
52.   }
53.
54.   function fetchWeather({
55.     city = 'Hanoi',
56.     country = 'Vietnam',
57.     units = 'metric'
58.   }) {
59.     return $.ajax(`${WEATHER_API_URL}&q=${city},${country}&units=${units}`);
60.   }
61.
62.   function renderWeather(weather) {
63.     const { icon, description } = weather.weather[0];
64.     const { temp, humidity, pressure } = weather.main;
65.     const { name } = weather;
66.
67.     $('#weather-detail').html(`
68.         <div>City: ${name}</div>
69.         <p class="text-capitalize">Weather: ${description}</p>
70.         <hr/>
71.         <div class="weather-detail-container">
72.           <div class="item toggle-units">
73.           <img class="img-lg" src="http://openweathermap.org/img/w/${icon}.png" />
74.           <span class="text-lg">${temp}</span>
75.           <div class="icon">
76.             <i class="wi wi-celsius icon-lg"/>
77.           </div>
78.         </div>
79.
80.         <div class="item">
81.           <p>Humidity: ${humidity} %</p>
82.           <p>Pressure: ${pressure} hPa</p>
83.         </div>`);
84.   }
85.
86.   ipLookUp();
87.
88.   // Declare DOM Object
89.   var form = $('#form');
90.   var input = $('#form-input');
91.   var $currentDate = $('#current-date');
92.
93.   // Handle user input
94.   form.on('submit', function(event) {
95.     event.preventDefault();
96.
97.     console.log('test', input.val());
98.     fetchWeather({ city: input.val().replace(/\s+/g, '') }).then(renderWeather);
99.   });
100.    });
101.
```

Step 13: Verify

Now, open **index.html** in Live Server and verify that all required functionalities is correctly implemented. You can try to search for Weather of: Ha Noi, Da Nang, Sai Gon and check the **Network** tab of Chrome

-- THE END --