

***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 |
|  | 25/Jun/2018 | Create a new Lab | Create new | DieuNT1 | VinhNV |
|  | 01/May/2019 | Update Fsoft Template | Update | DieuNT1 | VinhNV |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

Contents

[Unit 4 – jQuery and AJAX 4](#_Toc19527431)

[Objectives 4](#_Toc19527432)

[Technical Requirements: 4](#_Toc19527433)

[Specifications 4](#_Toc19527434)

[Guidelines 6](#_Toc19527435)

[Step 1: Create project structure 6](#_Toc19527436)

[Step 2: Open project in IDE 6](#_Toc19527437)

[Step 3: Create index page 7](#_Toc19527438)

[Step 4: Add Bootstrap 4 and Custom CSS 7](#_Toc19527439)

[Step 5: Add JavaScript 7](#_Toc19527440)

[Step 6: Create HTML layout 7](#_Toc19527441)

[Step 7: Declare DOM Objects 8](#_Toc19527442)

[Step 8: Bind Events 8](#_Toc19527443)

[Step 9: Get Location 8](#_Toc19527444)

[Step 10: Get Weather Information 9](#_Toc19527445)

[Step 11: Render Weather Information 9](#_Toc19527446)

[Step 12: Search for Weather Information from User input 10](#_Toc19527447)

[Step 13: Verify 12](#_Toc19527448)

|  |  |
| --- | --- |
|  | **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)

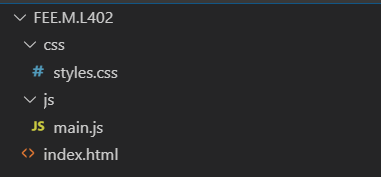
|  |
| --- |
|  |

* 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. <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css" integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO"
8. crossorigin="anonymous">
9. <link rel="stylesheet" type="text/css"
10. href="https://cdnjs.cloudflare.com/ajax/libs/weather-icons/2.0.9/css/weather-icons.min.css" />
11. <link rel="stylesheet" href="css/styles.css" media="screen" charset="utf-8">
12. </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. });

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. // Handle user input
7. form.on('submit', function(event) {
8. event.preventDefault();
9. console.log('test', input.val());
10. });
11. });

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

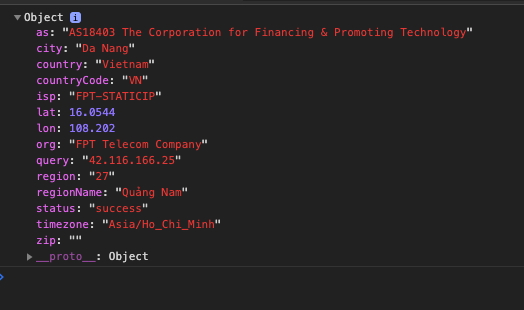
In the code above, we create 2 helper function that help us to format the current date.

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:



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:

1. function fetchWeather({
2. city = 'Hanoi',
3. country = 'Vietnam',
4. units = 'metric'
5. }) {
6. return $.ajax(`${WEATHER\_API\_URL}&q=${city},${country}&units=${units}`);
7. }

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

1. function renderWeather(weather) {
2. const { icon, description } = weather.weather[0];
3. const { temp, humidity, pressure } = weather.main;
4. const { name } = weather;
5. $('#weather-detail').html(`
6. <div>City: ${name}</div>
7. <p class="text-capitalize">Weather: ${description}</p>
8. <hr/>
9. <div class="weather-detail-container">
10. <div class="item toggle-units">
11. <img class="img-lg" src="http://openweathermap.org/img/w/${icon}.png" />
12. <span class="text-lg">${temp}</span>
13. <div class="icon">
14. <i class="wi wi-celsius icon-lg"/>
15. </div>
16. </div>
17. <div class="item">
18. <p>Humidity: ${humidity} %</p>
19. <p>Pressure: ${pressure} hPa</p>
20. </div>`);
21. }

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. function paddingLeft(n) {
5. return n < 10 ? `0${n}` : n;
6. }
7. function getCurrentDate() {
8. const days = [
9. 'Sunday',
10. 'Monday',
11. 'Tuesday',
12. 'Wednesday',
13. 'Thursday',
14. 'Friday',
15. 'Saturday'
16. ];
17. const months = [
18. 'Jan',
19. 'Feb',
20. 'Mar',
21. 'Apr',
22. 'May',
23. 'Jun',
24. 'Jul',
25. 'Aug',
26. 'Sep',
27. 'Oct',
28. 'Nov',
29. 'Dec'
30. ];
31. const now = new Date();
32. return `${
33. days[now.getDay()]
34. } ${months[now.getMonth()]} ${now.getDate()}, ${paddingLeft(now.getHours())}:${paddingLeft(now.getMinutes())}`;
35. }
36. function ipLookUp() {
37. $.ajax('http://ip-api.com/json').then(function success(response) {
38. $('#loading-location').hide();
39. $('#loading-weather').show();
40. console.log(response);
41. const { city, country } = response;
42. $currentDate.html(getCurrentDate());
43. fetchWeather({ city: city.replace(/\s+/g, ''), country }).then(renderWeather);
44. });
45. }
46. function fetchWeather({
47. city = 'Hanoi',
48. country = 'Vietnam',
49. units = 'metric'
50. }) {
51. return $.ajax(`${WEATHER\_API\_URL}&q=${city},${country}&units=${units}`);
52. }
53. function renderWeather(weather) {
54. const { icon, description } = weather.weather[0];
55. const { temp, humidity, pressure } = weather.main;
56. const { name } = weather;
57. $('#weather-detail').html(`
58. <div>City: ${name}</div>
59. <p class="text-capitalize">Weather: ${description}</p>
60. <hr/>
61. <div class="weather-detail-container">
62. <div class="item toggle-units">
63. <img class="img-lg" src="http://openweathermap.org/img/w/${icon}.png" />
64. <span class="text-lg">${temp}</span>
65. <div class="icon">
66. <i class="wi wi-celsius icon-lg"/>
67. </div>
68. </div>
69. <div class="item">
70. <p>Humidity: ${humidity} %</p>
71. <p>Pressure: ${pressure} hPa</p>
72. </div>`);
73. }
74. ipLookUp();
75. // Declare DOM Object
76. var form = $('#form');
77. var input = $('#form-input');
78. var $currentDate = $('#current-date');
79. // Handle user input
80. form.on('submit', function(event) {
81. event.preventDefault();
82. console.log('test', input.val());
83. fetchWeather({ city: input.val().replace(/\s+/g, '') }).then(renderWeather);
84. });
85. });

Step 13: Verify

Now, open **index.html** in Live Serverand 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 --**