

***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 3 – JavaScript 4](#_Toc19519509)

[Objectives 4](#_Toc19519510)

[Technical Requirements: 4](#_Toc19519511)

[Specifications 4](#_Toc19519512)

[Guidelines 5](#_Toc19519513)

[Step 1: Create project structure 5](#_Toc19519514)

[Step 2: Open project in IDE 5](#_Toc19519515)

[Step 3: Create index page 5](#_Toc19519516)

[Step 4: Add Bootstrap 4 and Custom CSS 5](#_Toc19519517)

[Step 5: Add JavaScript 6](#_Toc19519518)

[Step 6: Create HTML layout 6](#_Toc19519519)

[Step 7: Declare DOM Objects 6](#_Toc19519520)

[Step 8: Declare JavaScript Objects 7](#_Toc19519521)

[Step 9: Bind Events 7](#_Toc19519522)

[Step 10: Render TodoList 7](#_Toc19519523)

[Step 11: Implement Delete Button 8](#_Toc19519524)

[Step 12: Implement Complete Todo functionality 9](#_Toc19519525)

[Step 13: Verify 11](#_Toc19519526)

|  |  |
| --- | --- |
|  | **CODE: FEE.M.L301 (JavaScript)**  **TYPE: Medium**  **LOC: 200**  **DURATION: 90 MINUTES** |

# Unit 3 – JavaScript

Objectives

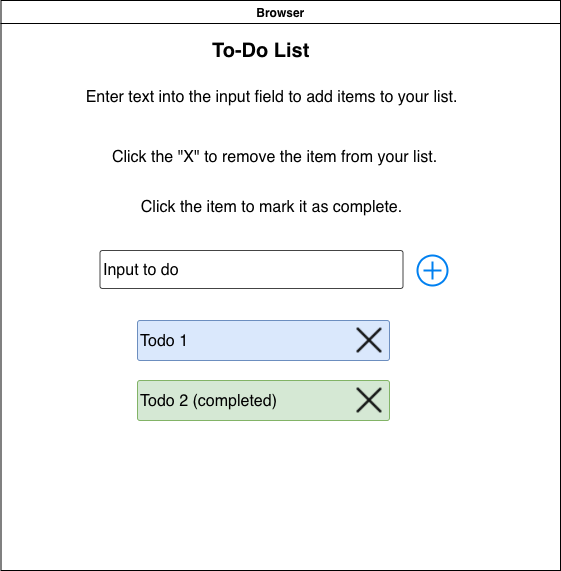
* Understand the core concepts of JavaScript programming language
* Understand basic concept of DOM
* Able to add behavior to make web site dynamic using JavaScript (DOM)
* Understand the core concepts of Bootstrap (layout, rows, grid, flex, components: buttons, alerts, utilities)

­­­­Technical Requirements:

* Must use HTML, CSS, and Bootstrap 4

Specifications

Your task is to implement a To-Do list web app like figure below:



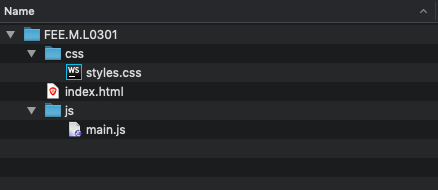
The app must fulfill below functionalities:

* User can type into an input text.
* When user press Enter, a new Todo task is added and the input text is cleared
* When user click on Add button (⊕ **symbol**), a new Todo task is added and the input text is cleared
* User can click on Delete button (**X symbol**) to remove a Todo
* User can click on a Todo to toggle its completeness i.e If a Todo is not complete, when user click on it will mark it as completed and the visual is changed so we can differentiate with other task.

Guidelines

Step 1: Create project structure

Create project structure like figure below:



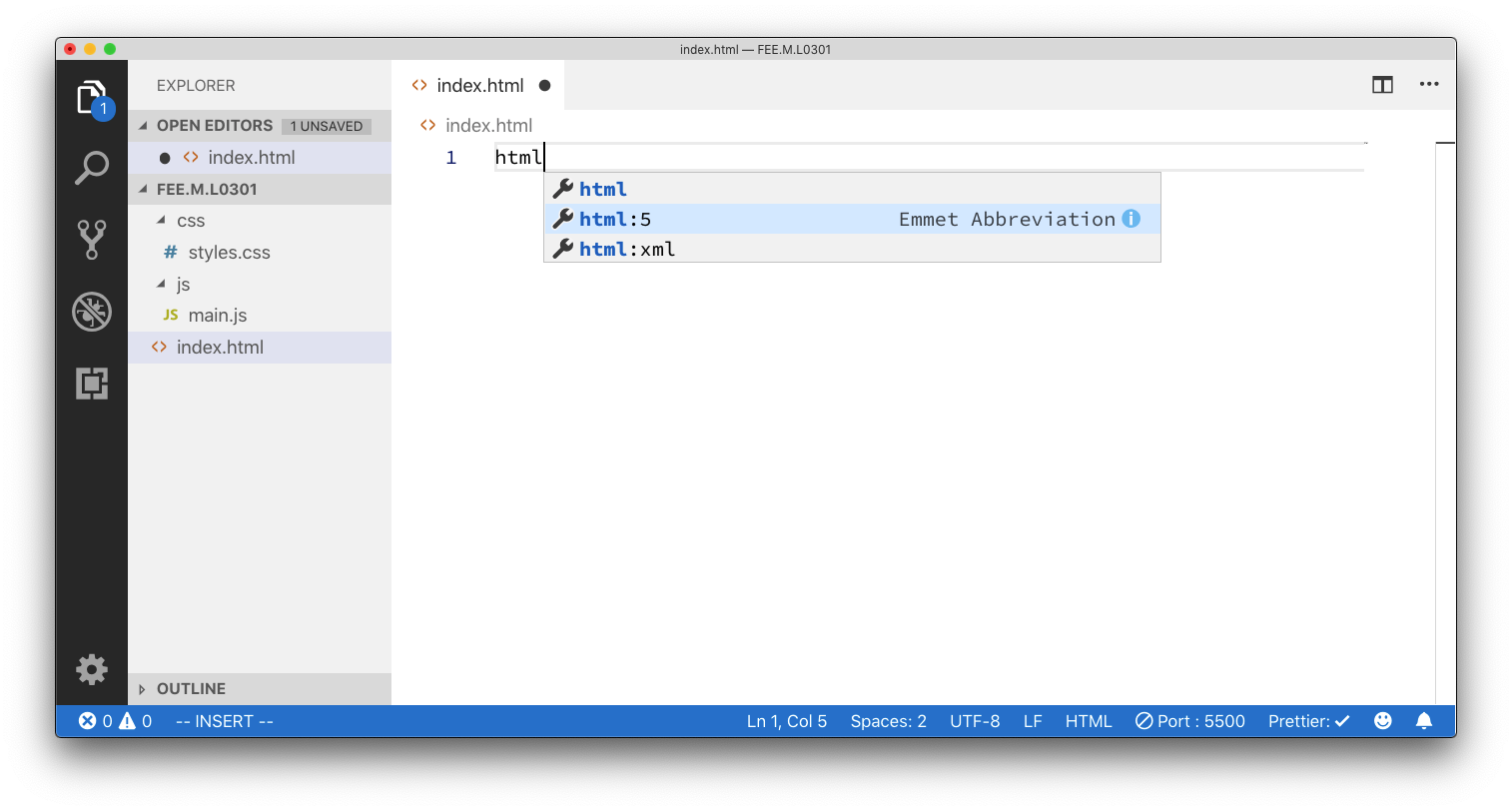
Step 2: Open project in IDE

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 **Todo List**

Step 4: Add Bootstrap 4 and Custom CSS

Add Bootstrap 4 CDN linkand **styles.css** file into head section of **index.html** file

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <meta charset="utf-8">
5. <title>Todo List</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" href="css/styles.css" media="screen" charset="utf-8">
10. </head>

Step 5: Add JavaScript

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

1. <body>
2. <script src="js/main.js"></script>
3. </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 class="container">
3. <h1 class="row justify-content-center">ToDo Lists</h1>
4. <h4 class="row justify-content-center">Enter text into the input field to add items to your list.</h4>
5. <h4 class="row justify-content-center">Click the item to mark it as complete.</h4>
6. <h4 class="row justify-content-center">Click the "X" to remove the item from your list.</h4>
7. <div class="row justify-content-center">
8. <section id="tasks">
9. <form id="tasks-form">
10. <div class="form-group d-flex">
11. <input type="text" class="form-control" placeholder="Enter Todo">
12. <button type="submit" class="btn btn-primary">Add</button>
13. </div>
14. </form>
15. <ul id="tasks-list">
16. </ul>
17. </section>
18. </div>
19. </div>
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. window.onload = function() {
2. // Declare DOM object
3. var container = document.querySelector('#tasks');
4. var form = container.querySelector('#tasks-form');
5. var taskText = form.querySelector('input');
6. var list = container.querySelector('#tasks-list');
7. };

Step 8: Declare JavaScript Objects

Next we need a variable to store a list of Todo

1. window.onload = function() {
2. // Declare DOM object
3. var container = document.querySelector('#tasks');
4. var form = container.querySelector('#tasks-form');
5. var taskText = form.querySelector('input');
6. var list = container.querySelector('#tasks-list');
7. // Declare JS Object
8. var todoList = [];
9. };

Step 9: 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. window.onload = function() {
2. // Declare DOM object
3. var container = document.querySelector('#tasks');
4. var form = container.querySelector('#tasks-form');
5. var taskText = form.querySelector('input');
6. var list = container.querySelector('#tasks-list');
7. // Declare JS Object
8. var todoList = [];
9. // Binding Events
10. form.addEventListener('submit', function(event) {
11. event.preventDefault();
12. // 1. Check if input text is empty
13. if (taskText.value == '') {
14. // Do nothing
15. return;
16. }
17. // 2. add a new Todo into the todoList
18. todoList.push({
19. id: new Date().getTime(),
20. text: taskText.value,
21. isCompleted: false
22. });
23. // 3. Clear input text
24. taskText.value = '';
25. // 4. render the todoList
26. renderTodoList();
27. });
28. };

Step 10: Render TodoList

To render TodoList, all we need to do is to loop the variable todoList and for each element in todoList, we will do the following:

1. Create a li element
2. Add CSS class to li element
3. Create a span to display the todo text value
4. Add span to li
5. Create a button (Delete button) and add to li
6. Add li to variable list (list is reference to <ul id=”tasks-list”>)

Check the code below:

1. window.onload = function() {
2. // Declare helper function
3. function renderTodoList() {
4. // clear list content
5. list.innerHTML = '';
6. todoList.forEach(todo => {
7. var li = document.createElement('li');
8. li.classList.add('list-group-item');
9. li.classList.add('d-flex');
10. li.classList.add('justify-content-between');
11. li.dataset.id = todo.id;
12. if (todo.isCompleted) {
13. li.classList.add('list-group-item-success');
14. }
15. var span = document.createElement('span');
16. span.textContent = todo.text;
17. var btnRemove = document.createElement('button');
18. btnRemove.dataset.id = todo.id;
19. btnRemove.textContent = 'Delete';
20. btnRemove.classList.add('btn');
21. btnRemove.classList.add('btn-danger');
22. li.appendChild(span);
23. li.appendChild(btnRemove);
24. list.appendChild(li);
25. });
26. }
27. // Declare DOM object
28. . . .
29. // Declare JS Object
30. . . .
31. // Binding Events
32. . . .
33. };

Open **index.html** in Live Server and try to add some Todo

Step 11: Implement Delete Button

Remember where the Delete Button is created?

To implement Delete functionality, we have to do the following:

1. Find the index of current selected Todo (based on dataset.id)
2. Remove Todo at index
3. Re-render TodoList by calling renderTodoList function
4. window.onload = function() {
5. // Declare helper function
6. function renderTodoList() {
7. // clear list content
8. list.innerHTML = '';
9. todoList.forEach(todo => {
10. var li = document.createElement('li');
11. li.classList.add('list-group-item');
12. li.classList.add('d-flex');
13. li.classList.add('justify-content-between');
14. li.dataset.id = todo.id;
15. if (todo.isCompleted) {
16. li.classList.add('list-group-item-success');
17. }
18. var span = document.createElement('span');
19. span.textContent = todo.text;
20. var btnRemove = document.createElement('button');
21. btnRemove.dataset.id = todo.id;
22. btnRemove.textContent = 'Delete';
23. btnRemove.classList.add('btn');
24. btnRemove.classList.add('btn-danger');
25. li.appendChild(span);
26. li.appendChild(btnRemove);
27. list.appendChild(li);
28. // Implement Delete button
29. btnRemove.removeEventListener('click', handleRemove);
30. btnRemove.addEventListener('click', handleRemove);
31. });
32. }
33. function handleRemove(event) {
34. event.stopPropagation();
35. // remove todo from todoList
36. var index = todoList.findIndex(t => t.id == this.dataset.id);
37. todoList.splice(index, 1);
38. // and re-render todoList
39. renderTodoList();
40. }
42. // Declare DOM object
43. . . .
44. // Declare JS Object
45. . . .
46. // Binding Events
47. . . .
48. };

Step 12: Implement Complete Todo functionality

Similarly to previous step, now when user click on the li item, then the selected item will be changed to completed (and vice versa). We do the same as previous step except we must bind on click event of li element

1. window.onload = function() {
2. // Declare helper function
3. function renderTodoList() {
4. // clear list content
5. list.innerHTML = '';
6. todoList.forEach(todo => {
7. var li = document.createElement('li');
8. li.classList.add('list-group-item');
9. li.classList.add('d-flex');
10. li.classList.add('justify-content-between');
11. li.dataset.id = todo.id;
12. if (todo.isCompleted) {
13. li.classList.add('list-group-item-success');
14. }
15. var span = document.createElement('span');
16. span.textContent = todo.text;
17. var btnRemove = document.createElement('button');
18. btnRemove.dataset.id = todo.id;
19. btnRemove.textContent = 'Delete';
20. btnRemove.classList.add('btn');
21. btnRemove.classList.add('btn-danger');
22. li.appendChild(span);
23. li.appendChild(btnRemove);
24. list.appendChild(li);
25. // Implement Delete button
26. btnRemove.removeEventListener('click', handleRemove);
27. btnRemove.addEventListener('click', handleRemove);
28. // Implement Complete functionality
29. li.removeEventListener('click', toggleTask);
30. li.addEventListener('click', toggleTask);
31. });
32. }
33. function handleRemove(event) {
34. event.stopPropagation();
35. // remove todo from todoList
36. var index = todoList.findIndex(t => t.id == this.dataset.id);
37. todoList.splice(index, 1);
38. // and re-render todoList
39. renderTodoList();
40. }
41. function toggleTask() {
42. // Find index of selected Todo
43. var index = todoList.findIndex(t => t.id == this.dataset.id);
44. // reverse current value of isCompleted
45. todoList[index].isCompleted = !todoList[index].isCompleted;
46. // re-render todoList
47. renderTodoList();
48. }
49. // Declare DOM object
50. var container = document.querySelector('#tasks');
51. var form = container.querySelector('#tasks-form');
52. var taskText = form.querySelector('input');
53. var list = container.querySelector('#tasks-list');
54. // Declare JS Object
55. var todoList = [];
56. // Binding Events
57. form.addEventListener('submit', function(event) {
58. event.preventDefault();
59. // 1. Check if input text is empty
60. if (taskText.value == '') {
61. // Do nothing
62. return;
63. }
64. // 2. add a new Todo into the todoList
65. todoList.push({
66. id: new Date().getTime(),
67. text: taskText.value,
68. isCompleted: false
69. });
70. // 3. Clear input text
71. taskText.value = '';
72. // 4. render the todoList
73. renderTodoList();
74. });
75. };

Step 13: Verify

Now, open **index.html** in Live Serverand verify that all required functionalities of Todo app is correctly implemented.

**-- THE END --**