Homework 3-1

Part 1:

Generally, I just move the texts in (<body><script></script></body>) into script.js.

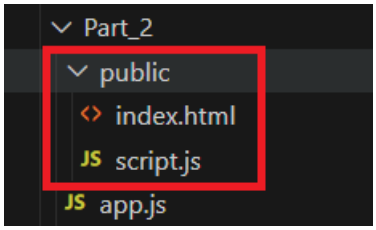
```
<body>
    <div class="container">
        <h1>Shopping List</h1>
        <input type="text" id="itemInput" placeholder="Add new item">
        <button onclick="addItem()">Add Item</button>
        <ul id="itemList"></ul>
    </div>
    <script>
        function addItem() {
            var input = document.getElementById("itemInput");
            var itemText = input.value.trim();
            if (itemText !== '') {
                var itemList = document.getElementById("itemList");
                var li = document.createElement("li");
                li.textContent = itemText;
                var deleteButton = document.createElement("button");
                deleteButton.textContent = "Delete";
                deleteButton.addEventListener("click", function() {
                    itemList.removeChild(li);
                });
                li.appendChild(deleteButton);
                itemList.appendChild(li);
                input.value = '';
            }
        }
    </script>
</body>
```

and link index.html and script.js using src attribute.

```
<script src="script.js"></script>
```

Part 2:

I put the index.html and script.js from Part 1 to a directory named public, (with some syntax modifications for the convenience of Part 3, but it's also alright to use the exact same code as part 1.)



and I create an app.js to serve static files from the 'public' directory and listen for incoming HTTP requests on port 3000.

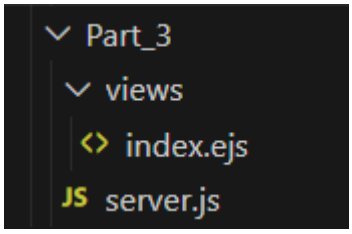
app.js

HW3_1 > Part_2 > app.js > ...

```
1   const express = require('express');
2   const app = express();
3   const port = 3000;
4
5   // 提供靜態資源
6   app.use(express.static('public'));
7
8   // 監聽端口
9   app.listen(port, () => {
10    console.log(`Shopping list app listening at http://localhost:${port}`);
11  });
12
```

Part 3:

I first create a directory called views with a file named index.ejs, and a server.js.



Add some basic components into the server.js:

```
server.js
HW3_1 > Part_3 > server.js > app.get('/') callback
1   const express = require('express');
2   const bodyParser = require('body-parser');
3   const app = express();
4   const port = 3000;
5
6   // 使用 body-parser 中間件來解析 POST 請求的正文
7   app.use(bodyParser.urlencoded({ extended: false }));
8
9   // 設置 EJS 模板引擎
10  app.set('view engine', 'ejs'); // 這代表 view engine 我們宣告為 ejs
11
12  // 購物清單數據
13  let items = [];
```

And in index.ejs, we create a submit form:

```
<form action="/addItem" method="post">
    <input type="text" name="newItem" placeholder="Add new item">
    <button type="submit">Add Item</button>
</form>
```

with its corresponding node.js part in server.js. This piece of code pushes the entered item into the newItem array:

```
// 添加新項目到購物清單
app.post('/addItem', (req, res) => {
  const newItem = req.body.newItem;
  if (newItem.trim() !== '') {
    items.push(newItem);
  }
  res.redirect('/');
});
```

Then I create a list with entered items and delete buttons. I use for loop to add new items and their corresponding delete buttons to the list:

```
<ul>
    <% for(let i=0; i<items.length; i++) { %>
        <li>
            <%= items[i] %>
            <form action="/deleteItem" method="post" style="display: inline;">
                <input type="hidden" name="index" value="<%= i %>">
                <button type="submit">Delete</button>
            </form>
        </li>
    <% } %>
</ul>
```

with its corresponding node.js part in server.js. This piece of code deletes one item from the item list:

```
// 刪除購物清單項目
app.post('/deleteItem', (req, res) => {
  const index = req.body.index;
  if (index >= 0 && index < items.length) {
    items.splice(index, 1);
  }
  res.redirect('/');
});
```

Lastly, we render the shopping list with the following code:

```
// 渲染購物清單頁面
app.get('/', (req, res) => {
  res.render('index', { items });
});
```

and listen for HTTP request:

```
// 監聽端口
app.listen(port, () => {
  console.log(`Shopping list app listening at http://localhost:${port}`);
});
```

Homework 3-2

Part 1:

Generally, I just move the texts in (<body><script></script></body>) into script.js.

```
<body>
    <div class="container">
        <h1>Weather App</h1>
        <input type="text" id="city-input" placeholder="Enter city name">
        <button id="search-button">Search</button>
        <div id="weather-info"></div>
    </div>

    <script>
        document.getElementById("search-button").addEventListener("click",
        function() {
            var city = document.getElementById("city-input").value;
            if (city.trim() !== "") {
                fetchWeather(city);
            } else {
                alert("Please enter a city name.");
            }
        });

        function fetchWeather(city) {
            var apiUrl = `https://wttr.in/${city}?format=%t+%w+%h`;

            fetch(apiUrl)
                .then(response => {
                    if (!response.ok) {
                        throw new Error("Failed to fetch weather data.");
                    }
                    return response.text();
                })
```

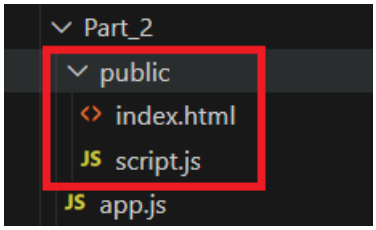
```
        .then(data => {
            displayWeather(data);
        })
        .catch(error => {
            console.error("Error fetching weather data:", error);
            alert("Failed to fetch weather data. Please try again later.");
        });
}

function displayWeather(data) {
    var weatherInfo = document.getElementById("weather-info");
    weatherInfo.innerText = data;
}
    </script>
</body>
```

and link index.html and script.js using src attribute.

```
<script src="script.js"></script>
```

Part 2:

I put the index.html and script.js from Part 1 to a directory named public, (with some syntax modifications for the convenience of Part 3, but it's also alright to use the exact same code as part 1.)



and I create an app.js to serve static files from the 'public' directory and listen for incoming HTTP requests on port 3000.

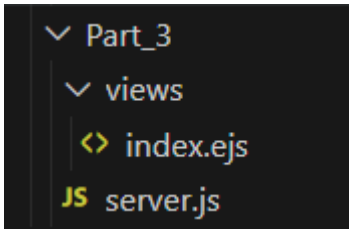
app.js

HW3_2 > Part_2 > app.js > ...

```
1   const express = require('express');
2   const app = express();
3   const port = 3000;
4
5   // 提供靜態資源
6   app.use(express.static('public'));
7
8   // 監聽端口
9   app.listen(port, () => {
10    console.log(`Weather app listening at http://localhost:${port}`);
11  });
12
```

Part 3:

I first create a directory called views with a file named index.ejs, and a server.js.



Add some basic components into the server.js:

```
server.js
HW3_2 > Part_3 > server.js > ...
1   const express = require('express');
2   const bodyParser = require('body-parser');
3   const app = express();
4   const port = 3000;
5
6   // 使用 body-parser 中間件來解析 POST 請求的正文
7   app.use(bodyParser.urlencoded({ extended: false }));
8
9   // 設置 EJS 模板引擎
10  app.set('view engine', 'ejs'); // 這代表 view engine 我們宣告為 ejs
11
12  let weatherInfo = "";
13  let alert = "";
```

And in index.ejs, compared to index.html in Part 2, I add the following two pieces:

```
<%= weatherInfo %>
```

```
<%- alert %>
```

with their corresponding node.js part in server.js. It's a post request that call the fetchWeather function:

```
// 處理天氣查詢請求
app.post('/citySubmit', (req, res) => {
    const city = req.body['city-input'];
    if (city.trim() !== "") {
        fetchWeather(city, res);
    } else {
        alert = '<script>alert("Please enter a city name.");</script>';
        res.redirect('/');
    }
});
```

And the function returns weather info from an outside api if available, pops up alerts if not.

```
function fetchWeather(city, res) {
    var apiUrl = `https://wttr.in/${city}?format=%t+%w+%h`;
    fetch(apiUrl)
        .then(response => {
            if (!response.ok) {
                throw new Error("Failed to fetch weather data.");
            }
            return response.text();
        })
        .then(data => {
            weatherInfo = data;
            res.redirect('/');
        })
        .catch(error => {
            console.error("Error fetching weather data:", error);
            alert = '<script>alert("Failed to fetch weather data. Please try again later.");</script>';
            res.redirect('/');
        });
}
```

Lastly, we render the webpage with the following code. Since the alert should only pop up once per error occurred, we add an currentAlert to save the current state of alert and clear the alert to “” when rendering:

```
15    // 渲染天氣資訊頁面
16    app.get('/', (req, res) => {
17        currentAlert = alert;
18        alert = "";
19        res.render('index', { weatherInfo, alert: currentAlert });
20      });
```

and listen for HTTP request:

```
// 監聽端口
app.listen(port, () => {
    console.log(`Weather app listening at http://localhost:${port}`);
});
```