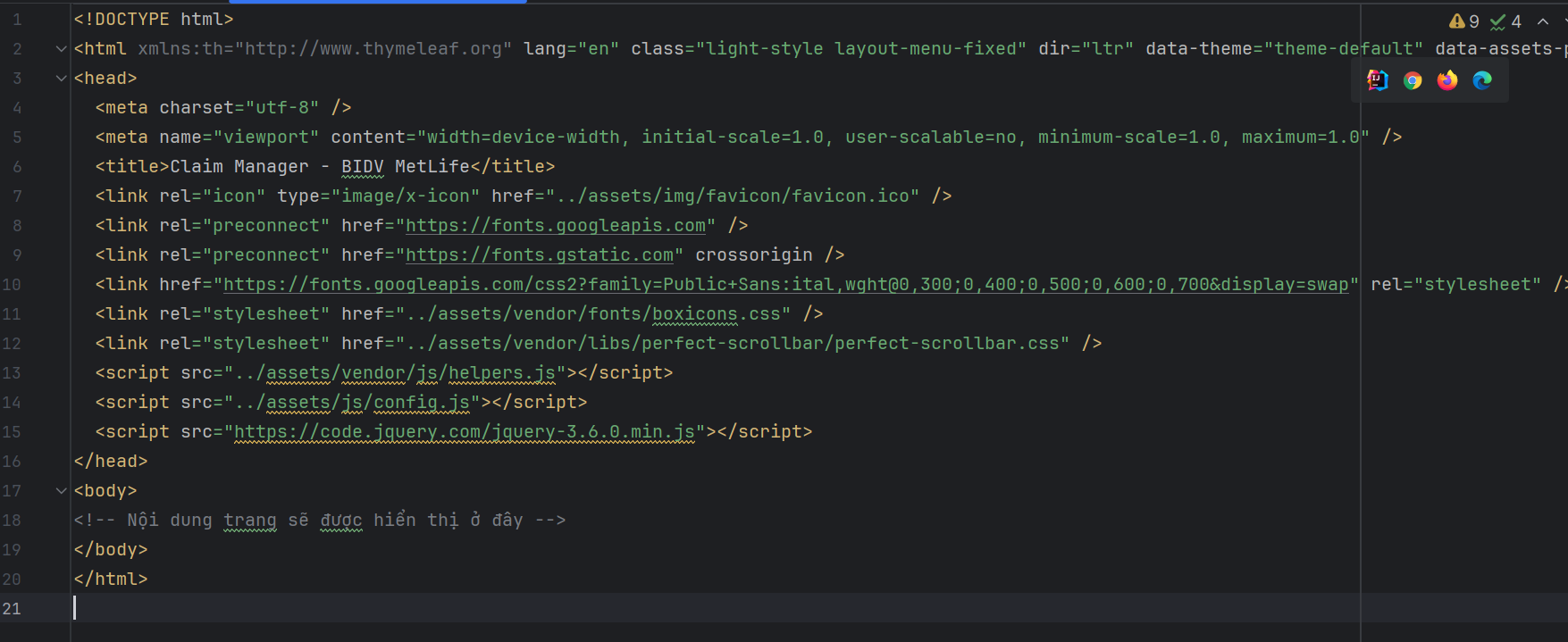
### 1. Kiến thức cơ bản về HTML, CSS và JavaScript

#### **HTML:**

1. **Cấu trúc cơ bản của một file HTML:** Một file HTML bao gồm các thẻ cơ bản như sau:
   * Thẻ <html>: Bọc toàn bộ nội dung của trang.
   * Thẻ <head>: Chứa thông tin không hiển thị trực tiếp trên trang như meta tags, liên kết CSS, và các file JavaScript.
   * Thẻ <body>: Chứa nội dung chính của trang web hiển thị cho người dùng như văn bản, hình ảnh, bảng, form.

**Ví dụ từ file claim-manager.html**:



* + <!DOCTYPE html> là định dạng cho trang HTML5.
  + Thẻ <meta charset="utf-8" /> xác định bộ ký tự cho trang, để đảm bảo mọi ký tự đặc biệt hiển thị đúng.
  + Thẻ <meta name="viewport" content="width=device-width, initial-scale=1.0" /> giúp trang thích nghi với các thiết bị di động.

1. **Cách tạo các thẻ input form (input text, select box, button):** Thẻ form dùng để nhận thông tin từ người dùng. Các thẻ cơ bản bao gồm:
   * <input type="text"> để tạo ô nhập văn bản.
   * <select> để tạo hộp lựa chọn (dropdown).
   * <button> để tạo nút nhấn.

**Ví dụ từ file claim-manager.html**:

html

<div class="row mb-3">  
 <div class="col-md-3">  
 <label for="search-ma-yeu-cau">Mã yêu cầu bồi thường</label>  
 <input type="text" id="search-ma-yeu-cau" class="form-control" placeholder="Nhập mã yêu cầu">  
 </div>  
 <div class="col-md-3">  
 <label for="from-date">Từ ngày</label>  
 <input type="date" id="from-date" class="form-control">  
 </div>  
 <div class="col-md-3">  
 <label for="to-date">Đến ngày</label>  
 <input type="date" id="to-date" class="form-control">  
 </div>  
 <div class="col-md-3">  
 <label for="trang-thai-yeu-cau">Trạng thái yêu cầu</label>  
 <select id="trang-thai-yeu-cau" class="form-control">  
 <option value="">Tất cả trạng thái</option>  
 <option value="Đang xử lý">Đang xử lý</option>  
 <option value="Hoàn thành">Hoàn thành</option>  
 <option value="Đã lên lịch">Đã lên lịch</option>  
 <option value="Đang chờ">Đang chờ</option>  
 <option value="Bị từ chối">Bị từ chối</option>  
 </select>  
 </div>  
</div>  
<div class="row mb-3">  
 <div class="col-md-12 text-right">  
 <button id="search-button" class="btn btn-primary">Tìm kiếm</button>  
 </div>  
</div>

</div>

* + input type="text" tạo một ô để người dùng nhập văn bản.
  + select tạo một danh sách thả xuống với các lựa chọn.
  + button tạo một nút nhấn để thực hiện hành động, trong trường hợp này là nút "Tìm kiếm".

1. **Cách tạo bảng dữ liệu (<table>) để hiển thị thông tin danh sách:** Bảng (<table>) được dùng để hiển thị dữ liệu dưới dạng lưới. Nó bao gồm các thẻ:
   * <thead>: Đầu bảng (tiêu đề các cột).
   * <tbody>: Phần nội dung chính (dữ liệu).
   * <tr>: Mỗi hàng trong bảng.
   * <td>: Mỗi ô dữ liệu.

**Ví dụ từ file claim-manager.html**:

<table class="table" id="claims-table">  
 <thead>  
 <tr>  
 <th><input type="checkbox" id="selectAll"></th>  
 <th>Mã yêu cầu bồi thường</th>  
 <th>Tên khách hàng</th>  
 <th>Tên sản phẩm bảo hiểm</th>  
 <th>Ngày gửi yêu cầu</th>  
 <th>Mô tả yêu cầu</th>  
 <th>Trạng thái yêu cầu</th>  
 </tr>  
 </thead>  
 <tbody class="table-border-bottom-0">  
 <tr>  
 <td><input type="checkbox"></td>  
 <td>YC001</td>  
 <td>Nguyễn Văn A</td>  
 <td>Bảo hiểm xe máy</td>  
 <td>2024-09-01</td>  
 <td>Hư hại xe sau tai nạn</td>  
 <td>Đang xử lý</td>  
 </tr>  
 <tr>  
 <td><input type="checkbox"></td>  
 <td>YC002</td>  
 <td>Trần Thị B</td>  
 <td>Bảo hiểm y tế</td>  
 <td>2024-08-25</td>  
 <td>Viện phí sau phẫu thuật</td>  
 <td>Đã giải quyết</td>  
 </tr>  
 </tbody>  
</table>

#### **CSS:**

1. **Sử dụng các class để định nghĩa giao diện cho các phần tử HTML:** Mỗi phần tử HTML có thể có một hoặc nhiều class để xác định kiểu dáng của nó, được định nghĩa trong file CSS.

**Ví dụ từ file claim-manager.html**:

html

<div class="col-md-3">  
 <label for="search-ma-yeu-cau">Mã yêu cầu bồi thường</label>  
 <input type="text" id="search-ma-yeu-cau" class="form-control" placeholder="Nhập mã yêu cầu">  
</div>

Ở đây, class form-control có thể được định nghĩa trong file CSS để làm cho ô nhập liệu có kích thước và giao diện đồng nhất.

1. **Liên kết file CSS từ bên ngoài:** Bạn có thể sử dụng các file CSS bên ngoài để định nghĩa giao diện của trang web bằng cách liên kết chúng qua thẻ <link>.

**Ví dụ từ file claim-manager.html**:

<link rel="stylesheet" href="../assets/vendor/fonts/boxicons.css" />  
<link rel="stylesheet" href="../assets/vendor/css/core.css" class="template-customizer-core-css" />  
<link rel="stylesheet" href="../assets/vendor/css/theme-default.css" class="template-customizer-theme-css" />

Điều này giúp tải các tệp CSS từ bên ngoài vào để áp dụng các kiểu dáng lên trang web.

1. **Sử dụng thư viện CSS có sẵn:** Bạn có thể sử dụng các thư viện CSS có sẵn để nhanh chóng áp dụng giao diện đẹp mắt mà không cần tự viết CSS từ đầu.

**Ví dụ từ file claim-manager.html**:

<link href="https://fonts.googleapis.com/css2?family=Public+Sans:ital,wght@0,300;0,400;0,500;0,600;0,700&display=swap" rel="stylesheet" />  
<link rel="stylesheet" href="../assets/vendor/fonts/boxicons.css" />  
<link rel="stylesheet" href="../assets/vendor/css/core.css" class="template-customizer-core-css" />

<link rel="stylesheet" href="../assets/vendor/css/core.css" />

* + boxicons.css là thư viện biểu tượng.
  + core.css là các tệp giao diện cơ bản cho trang web.

#### **JavaScript:**

1. **Khái niệm về DOM (Document Object Model):** DOM là cách trình duyệt hiểu và truy cập các phần tử HTML trên trang web. JavaScript cho phép bạn thao tác các phần tử trên trang, thay đổi nội dung hoặc kiểu dáng.

**Ví dụ**: DOM giúp bạn truy cập và thay đổi giá trị của một ô nhập văn bản như sau:

javascript

const claimCode = $('#search-ma-yeu-cau').val();  
const fromDate = $('#from-date').val();  
const toDate = $('#to-date').val();  
const statusCode = $('#trang-thai-yeu-cau').val();

các câu lệnh trên giúp lấy giá trị tại các ô input của trang web

1. **Cách lắng nghe sự kiện (sử dụng jQuery):** Bạn có thể lắng nghe sự kiện từ người dùng, ví dụ khi người dùng nhấn nút, thay đổi giá trị của ô nhập, hoặc chọn một lựa chọn.

**Ví dụ từ file claim-manager.js**:

javascript

$('#search-button').click(function() {  
 fetchClaims();  
});

1. **Cách sử dụng các chức năng cơ bản của jQuery như .val(), .append(), .empty():** jQuery cung cấp các hàm tiện lợi để thao tác với DOM.
   * .val(): Lấy hoặc đặt giá trị của một ô nhập liệu.
   * .append(): Thêm một phần tử mới vào bên trong một phần tử khác.
   * .empty(): Xóa sạch nội dung bên trong một phần tử.

**Ví dụ từ file claim-manager.js**:

javascript

function renderTable(data) {  
 const tbody = $('#claims-table tbody');  
 tbody.empty();  
 data.content.forEach(item => {  
 const row = `<tr>  
 <td><input type="checkbox" class="recordCheckbox"></td>  
 <td><a href="detail-claim.html" style="text-decoration: none;"><strong>${item.claimCode}</strong></a></td>  
 <td>${item.customerName}</td>  
 <td>${item.insuranceProductName}</td>  
 <td>${item.claimDate}</td>  
 <td>${item.description}</td>  
 <td><span class="badge bg-label-primary me-1">${item.statusDescription}</span></td>  
 </tr>`;  
 tbody.append(row);  
 });  
}

### Kết luận:

* **HTML**: Xây dựng cấu trúc cơ bản của trang web, bao gồm form nhập liệu và bảng dữ liệu.
* **CSS**: Định nghĩa giao diện cho các phần tử bằng cách sử dụng class và các thư viện có sẵn.
* **JavaScript (jQuery)**: Tương tác với các phần tử trang web thông qua DOM, lắng nghe sự kiện và thao tác dữ liệu trên giao diện.

### 2. Kiến thức về jQuery

#### **Khái niệm cơ bản về jQuery**

**jQuery** là một thư viện JavaScript giúp đơn giản hóa các thao tác phổ biến trên trang web như:

* Truy cập và thay đổi các phần tử HTML.
* Xử lý sự kiện người dùng (như click, nhập liệu).
* Gửi và nhận dữ liệu từ server (AJAX).

jQuery được sử dụng rộng rãi vì nó giúp mã JavaScript ngắn gọn hơn và dễ viết hơn.

**Cách tích hợp jQuery trong dự án:**

Để sử dụng jQuery trong một dự án, bạn có thể tích hợp thông qua CDN (Content Delivery Network). Chỉ cần thêm đoạn mã sau vào file HTML, thường nằm trong phần <head> hoặc cuối phần <body>:

html

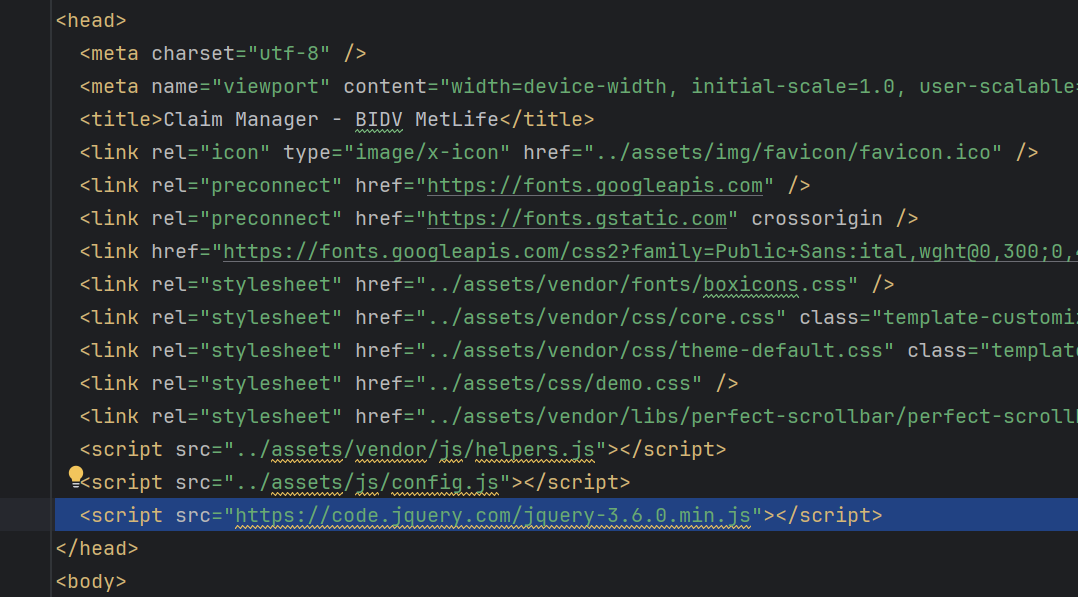
Sao chép mã

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

**Ví dụ**:

Trong file claim-manager.html, jQuery được tích hợp như sau:

html

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

Sau khi tích hợp jQuery, bạn có thể sử dụng cú pháp $() để truy cập và thao tác với các phần tử trên trang web.

**Ví dụ**:

javascript

Sao chép mã

$('#button').click(function() {  
 alert('Nút đã được nhấn!');  
});

#### **AJAX với jQuery**

**AJAX** (Asynchronous JavaScript and XML) là kỹ thuật cho phép gửi và nhận dữ liệu từ server mà không cần tải lại toàn bộ trang. Trong jQuery, bạn có thể dễ dàng thực hiện AJAX bằng phương thức $.ajax().

##### **Cách gửi yêu cầu AJAX tới server (POST dữ liệu)**

Phương thức POST được sử dụng để gửi dữ liệu từ trình duyệt lên server. Trong jQuery, bạn có thể sử dụng $.ajax() để gửi yêu cầu POST với cú pháp như sau:

javascript

$.ajax({  
 url: 'URL của API', // Địa chỉ API mà bạn muốn gửi dữ liệu  
 type: 'POST', // Phương thức POST  
 dataType: 'json', // Kiểu dữ liệu nhận về  
 contentType: 'application/json', // Kiểu dữ liệu gửi đi  
 data: *JSON*.stringify({ key1: 'value1', key2: 'value2' }), // Dữ liệu được gửi  
 success: function(response) {  
 // Xử lý khi thành công  
 *console*.log(response);  
 },  
 error: function(error) {  
 // Xử lý khi xảy ra lỗi  
 *console*.error('Lỗi:', error);  
 }  
});

**Ví dụ từ file claim-manager.js:**

Trong file này, yêu cầu AJAX được gửi tới API với phương thức POST để lấy danh sách yêu cầu bồi thường:

javascript

function fetchClaims(page = 0) {  
 const claimCode = $('#search-ma-yeu-cau').val();  
 const fromDate = $('#from-date').val();  
 const toDate = $('#to-date').val();  
 const statusCode = $('#trang-thai-yeu-cau').val();  
  
 const requestData = {  
 claimCode,  
 fromDate,  
 toDate,  
 statusCode  
 };  
  
 $.ajax({  
 url: `http://localhost:8080/api/claims?page=${page}&size=`+PAGE\_SIZE,  
 type: 'POST',  
 contentType: 'application/json',  
 data: *JSON*.stringify(requestData),  
 success: function(response) {  
 renderTable(response);  
 renderPagination(response.pageNumber, response.totalPages);  
 },  
 error: function(error) {  
 *console*.error('Error fetching claims:', error);  
 }  
 });  
}

##### **Cách xử lý dữ liệu JSON trả về từ server và hiển thị trên giao diện người dùng**

Sau khi nhận được phản hồi từ server (thường ở định dạng JSON), bạn có thể sử dụng dữ liệu này để cập nhật giao diện người dùng.

**Ví dụ từ file claim-manager.js:**

Dữ liệu trả về từ server được hiển thị trong bảng HTML bằng cách sử dụng vòng lặp:

javascript

Sao chép mã

function renderTable(data) {  
 const tbody = $('#claims-table tbody');  
 tbody.empty();  
 data.content.forEach(item => {  
 const row = `<tr>  
 <td><input type="checkbox" class="recordCheckbox"></td>  
 <td><a href="detail-claim.html" style="text-decoration: none;"><strong>${item.claimCode}</strong></a></td>  
 <td>${item.customerName}</td>  
 <td>${item.insuranceProductName}</td>  
 <td>${item.claimDate}</td>  
 <td>${item.description}</td>  
 <td><span class="badge bg-label-primary me-1">${item.statusDescription}</span></td>  
 </tr>`;  
 tbody.append(row);  
 });  
}

}

**Giải thích**:

* Hàm renderTable nhận dữ liệu JSON từ server, trong đó mỗi mục (item) chứa thông tin về một yêu cầu bồi thường.
* Vòng lặp forEach được sử dụng để duyệt qua từng mục trong dữ liệu và tạo ra một dòng (row) mới trong bảng HTML.
* Dữ liệu từ server (như claimCode, customerName, v.v.) được chèn trực tiếp vào các phần tử HTML.

##### **Xử lý lỗi trong AJAX**

Nếu yêu cầu AJAX gặp lỗi (ví dụ: không thể kết nối đến server), hàm error sẽ được gọi. Trong ví dụ của claim-manager.js, lỗi được xử lý như sau:

$.ajax({  
 url: `http://localhost:8080/api/claims?page=${page}&size=3`,  
 type: 'POST',  
 contentType: 'application/json',  
 data: *JSON*.stringify(requestData),  
 success: function(response) {  
 renderTable(response);  
 renderPagination(response.pageNumber, response.totalPages);  
 },  
 error: function(error) {  
 *console*.error('Lỗi khi lấy dữ liệu:', error); // In lỗi ra console  
 }  
});

Khi lỗi xảy ra, dòng console.error sẽ in chi tiết lỗi ra console của trình duyệt. Điều này giúp nhà phát triển dễ dàng kiểm tra và khắc phục lỗi.

### ****Kết luận:****

* **jQuery** giúp đơn giản hóa các thao tác với HTML, đặc biệt là xử lý sự kiện và AJAX.
* **AJAX** với jQuery cho phép gửi yêu cầu tới server và nhận dữ liệu mà không cần tải lại trang.
* Phương thức POST được sử dụng để gửi dữ liệu và $.ajax() là cách phổ biến để thực hiện AJAX trong jQuery.
* Bạn có thể xử lý dữ liệu JSON trả về từ server và hiển thị nó lên giao diện người dùng thông qua các hàm như renderTable.
* Khi xảy ra lỗi trong quá trình gửi yêu cầu AJAX, bạn có thể dễ dàng bắt lỗi và xử lý thông qua hàm error.

### ****3 Gắn sự kiện trong jQuery****

Một trong những tính năng mạnh mẽ của jQuery là xử lý sự kiện. Trong jQuery, bạn có thể gắn sự kiện (event) cho các phần tử HTML một cách dễ dàng, giúp trang web tương tác với người dùng thông qua các hành động như nhấn nút, di chuyển chuột, nhập liệu, và nhiều hơn nữa.

#### **Khái niệm về sự kiện (Event)**

Sự kiện là những hành động mà người dùng thực hiện trên trang web, như:

* Nhấn nút (click).
* Nhập liệu vào ô input.
* Di chuyển chuột (hover).
* Chọn mục từ danh sách thả xuống.

jQuery cung cấp các phương thức để gắn sự kiện vào các phần tử HTML, giúp trang web phản hồi lại các hành động của người dùng.

#### **Gắn sự kiện (Event Binding) trong jQuery**

Dưới đây là các sự kiện phổ biến và cách gắn chúng vào các phần tử HTML bằng jQuery:

1. **Sự kiện click:**

Sự kiện click được gắn khi người dùng nhấn vào một phần tử, như nút (<button>) hoặc liên kết (<a>).

**Ví dụ**:

javascript

// Gắn sự kiện click vào nút có id là "myButton"

$('#myButton').click(function() {  
 alert('Nút đã được nhấn!');  
});

Trong ví dụ trên, khi người dùng nhấn vào nút có id là myButton, sẽ có một cửa sổ thông báo xuất hiện với thông điệp "Nút đã được nhấn!".

1. **Sự kiện change:**

Sự kiện change được sử dụng để xử lý khi người dùng thay đổi giá trị trong các phần tử input như hộp chọn (<select>) hoặc ô nhập (<input>).

**Ví dụ**:

javascript

// Gắn sự kiện change vào phần tử select có id là "mySelect"

$('#mySelect').change(function() {  
 alert('Bạn đã chọn: ' + $(this).val());  
});

Trong ví dụ trên, khi người dùng chọn một tùy chọn khác từ danh sách select, một thông báo sẽ hiển thị giá trị mới mà người dùng đã chọn.

1. **Sự kiện keyup:**

Sự kiện keyup được kích hoạt khi người dùng nhả phím sau khi nhấn xuống trong ô input.

**Ví dụ**:

javascript

// Gắn sự kiện keyup vào ô nhập có id là "textInput"

$('#textInput').keyup(function() {  
 *console*.log('Bạn vừa nhập: ' + $(this).val());  
});

Trong ví dụ này, mỗi lần người dùng nhập ký tự vào ô input textInput, giá trị của ô sẽ được hiển thị trong console.

1. **Sự kiện hover:**

Sự kiện hover được sử dụng để xử lý khi người dùng di chuyển chuột vào và ra khỏi một phần tử.

**Ví dụ**:

javascript

// Gắn sự kiện hover vào phần tử có id là "hoverBox"

$('#hoverBox').hover(  
 function() { // Khi di chuột vào  
 $(this).css('background-color', 'yellow');  
 },  
 function() { // Khi di chuột ra  
 $(this).css('background-color', 'white');  
 }  
);

Trong ví dụ trên, khi người dùng di chuyển chuột vào phần tử hoverBox, nền của nó sẽ chuyển sang màu vàng. Khi chuột di chuyển ra ngoài, nền sẽ trở lại màu trắng.

1. **Sự kiện submit:**

Sự kiện submit được gắn vào các form và kích hoạt khi người dùng nhấn nút submit để gửi form.

**Ví dụ**:

javascript

// Gắn sự kiện submit vào form có id là "myForm"

$('#myForm').submit(function(e) {  
 e.preventDefault(); // Ngăn chặn form không bị gửi đi  
 alert('Form đã được submit!');  
});

Trong ví dụ trên, khi người dùng nhấn nút submit, sự kiện submit được kích hoạt, nhưng với e.preventDefault(), việc gửi form sẽ bị chặn, và thay vào đó một thông báo sẽ hiện ra.

#### **Phương thức** .on() **trong jQuery**

Phương thức .on() là cách gắn sự kiện phổ biến và linh hoạt nhất trong jQuery. Nó cho phép bạn gắn nhiều sự kiện vào một phần tử hoặc gắn sự kiện cho các phần tử động (các phần tử được tạo ra sau khi trang đã tải).

**Ví dụ**:

$('.myButton').on('click', function() {  
 alert('Nút có class "myButton" đã được nhấn!');  
});

Bạn có thể gắn nhiều sự kiện vào một phần tử bằng cách sử dụng .on():

$('#myElement').on({  
 click: function() {  
 alert('Đã click!');  
 },  
 mouseenter: function() {  
 $(this).css('background-color', 'lightblue');  
 },  
 mouseleave: function() {  
 $(this).css('background-color', 'white');  
 }  
});

Trong ví dụ trên:

* Khi nhấn vào phần tử #myElement, sẽ có thông báo hiện ra.
* Khi di chuột vào phần tử, nền sẽ đổi thành màu xanh nhạt.
* Khi di chuột ra khỏi phần tử, nền sẽ trở lại màu trắng.

#### **Ví dụ từ file** claim-manager.js**:**

Trong file này, sự kiện click được gắn vào nút #search-button để thực hiện tìm kiếm khi người dùng nhấn vào nút:

javascript

Sao chép mã

$('#search-button').click(function() {  
 fetchClaims();  
});

Ngoài ra, sự kiện click cũng được gắn vào các nút phân trang động, giúp người dùng chuyển đổi giữa các trang dữ liệu:

javascript

$(*document*).on('click', '#pagination .page-link', function(e) {  
 e.preventDefault(); // Ngăn chặn hành động mặc định của link  
 const page = $(this).data('page'); // Lấy số trang từ thuộc tính data-page  
 fetchClaims(page); // Gọi hàm fetchClaims với trang mới  
});

### ****Kết luận:****

* **Sự kiện (Event)** là những hành động mà người dùng thực hiện trên trang web, như nhấn nút, nhập liệu hoặc di chuyển chuột.
* jQuery giúp việc gắn sự kiện trở nên dễ dàng hơn với cú pháp ngắn gọn như .click(), .change(), .keyup(), .hover(), và .submit().
* Phương thức .on() cung cấp cách gắn nhiều sự kiện cho một phần tử hoặc gắn sự kiện cho các phần tử được tạo ra động.

### 4. Tạo và quản lý bảng danh sách với dữ liệu động

Trong phần này, chúng ta sẽ tìm hiểu cách tạo bảng HTML và sử dụng JavaScript để quản lý dữ liệu động. Đây là một kỹ thuật rất quan trọng khi xây dựng các ứng dụng web hiện đại, giúp dữ liệu hiển thị trên trang web được cập nhật liên tục mà không cần tải lại trang.

### ****Tạo bảng HTML****

Bảng HTML là một cấu trúc hữu ích để hiển thị dữ liệu dưới dạng lưới, thường được sử dụng trong các ứng dụng quản lý danh sách, như danh sách sản phẩm, danh sách yêu cầu, v.v. Bảng HTML được tạo ra với các thẻ cơ bản như <table>, <thead>, <tbody>, <tr>, và <td>.

#### **1. Cấu trúc cơ bản của bảng HTML**

**Các thẻ cơ bản trong bảng HTML**:

* <table>: Thẻ chính bao bọc toàn bộ bảng.
* <thead>: Chứa các tiêu đề cột.
* <tbody>: Chứa nội dung chính của bảng (các dòng dữ liệu).
* <tr>: Đại diện cho mỗi hàng trong bảng.
* <td>: Đại diện cho từng ô trong mỗi hàng.

**Ví dụ** về một bảng HTML hiển thị danh sách yêu cầu bồi thường:

html

<table class="table" id="claims-table">  
 <thead>  
 <tr>  
 <th><input type="checkbox" id="selectAll"></th>  
 <th>Mã yêu cầu bồi thường</th>  
 <th>Tên khách hàng</th>  
 <th>Tên sản phẩm bảo hiểm</th>  
 <th>Ngày gửi yêu cầu</th>  
 <th>Mô tả yêu cầu</th>  
 <th>Trạng thái yêu cầu</th>  
 </tr>  
 </thead>  
 <tbody class="table-border-bottom-0">  
 <!-- Các hàng dữ liệu sẽ được thêm động ở đây -->  
 </tbody>  
</table>

#### **2. Đưa dữ liệu động vào bảng HTML bằng JavaScript**

Dữ liệu từ server thường được trả về dưới dạng JSON. Bạn có thể sử dụng JavaScript để xử lý dữ liệu này và đưa vào bảng HTML. Việc này thường được thực hiện bằng cách sử dụng vòng lặp để duyệt qua danh sách dữ liệu và tạo các hàng tương ứng.

**Ví dụ**: Trong file claim-manager.js, hàm renderTable được sử dụng để hiển thị dữ liệu vào bảng.

javascript

function renderTable(data) {  
 const tbody = $('#claims-table tbody');  
 tbody.empty();  
 data.content.forEach(item => {  
 const row = `<tr>  
 <td><input type="checkbox" class="recordCheckbox"></td>  
 <td><a href="detail-claim.html" style="text-decoration: none;"><strong>${item.claimCode}</strong></a></td>  
 <td>${item.customerName}</td>  
 <td>${item.insuranceProductName}</td>  
 <td>${item.claimDate}</td>  
 <td>${item.description}</td>  
 <td><span class="badge bg-label-primary me-1">${item.statusDescription}</span></td>  
 </tr>`;  
 tbody.append(row);  
 });  
}

**Giải thích**:

* Hàm renderTable nhận dữ liệu JSON từ server và tạo các hàng (<tr>) trong bảng HTML.
* Dữ liệu được chèn vào từng cột (<td>) trong mỗi hàng bằng cú pháp JavaScript.
* tbody.empty() đảm bảo rằng bảng được xóa sạch trước khi thêm dữ liệu mới, tránh tình trạng hiển thị dữ liệu trùng lặp.

### ****Sử dụng checkbox và nút chức năng****

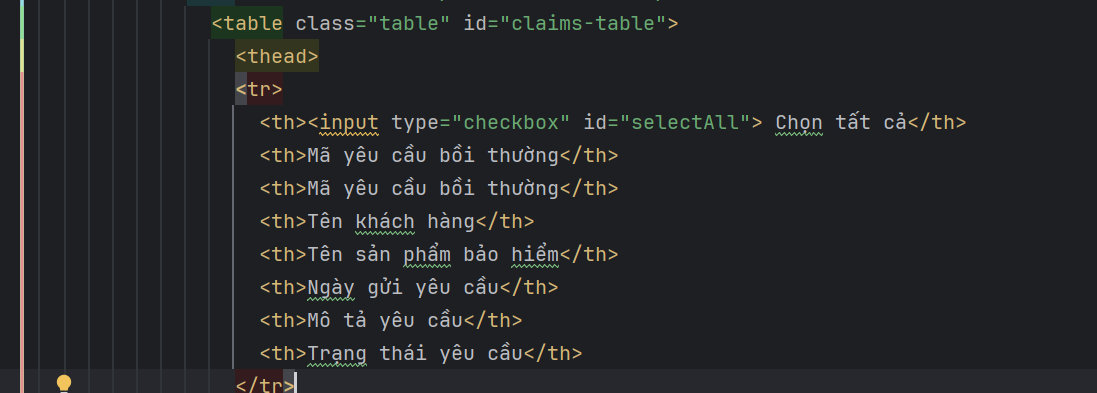
Các checkbox và nút chức năng thường được tích hợp vào bảng HTML để người dùng có thể chọn các hàng, sau đó thực hiện các hành động như "Chấp nhận", "Từ chối", hoặc "Xóa".

#### **1. Checkbox để chọn các hàng trong bảng**

Checkbox thường được sử dụng trong bảng HTML để cho phép người dùng chọn một hoặc nhiều hàng dữ liệu. Bạn có thể tích hợp checkbox trong mỗi hàng bằng cách thêm <input type="checkbox"> vào các cột tương ứng.

**Ví dụ** về tích hợp checkbox trong bảng:

html

Trong ví dụ trên:

* Checkbox "Chọn tất cả" (id="selectAll") nằm ở tiêu đề bảng và có thể được sử dụng để chọn hoặc bỏ chọn tất cả các hàng.

**Tích hợp sự kiện để chọn tất cả hàng**:

javascript

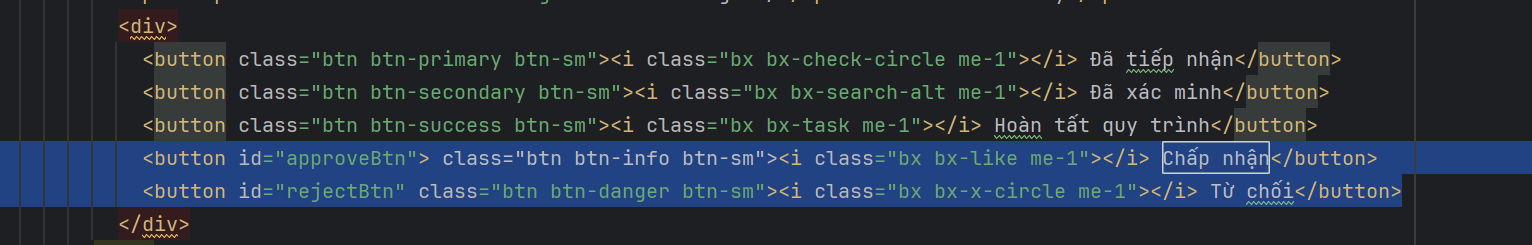
$('#selectAll').click(function() {  
 $('.recordCheckbox').prop('checked', this.checked); // Chọn hoặc bỏ chọn tất cả checkbox trong bảng  
});

#### **2. Thêm các nút chức năng để điều khiển trạng thái yêu cầu**

Các nút chức năng có thể được sử dụng để thực hiện các hành động như "Chấp nhận", "Từ chối", "Hoàn tất", v.v. Những nút này có thể nằm phía trên bảng hoặc ở cột cuối cùng của mỗi hàng.

**Ví dụ** trong file HTML:

html

**Xử lý sự kiện cho nút chức năng**: Khi người dùng nhấn vào một nút, bạn có thể thực hiện các hành động tương ứng. Ví dụ, khi người dùng chọn các hàng và nhấn "Chấp nhận", bạn có thể lấy danh sách các hàng đã chọn và gửi yêu cầu xử lý tới server.

javascript

Sao chép mã

$('#approveBtn').click(function() {  
 let selectedClaims = [];  
 $('.recordCheckbox:checked').each(function() {  
 selectedClaims.push($(this).closest('tr').find('td:eq(1)').text()); // Lấy mã yêu cầu từ các hàng đã chọn  
 });  
  
 if (selectedClaims.length > 0) {  
 // Gửi dữ liệu đã chọn lên server hoặc thực hiện hành động  
 *console*.log('Chấp nhận các yêu cầu: ' + selectedClaims.join(', '));  
 } else {  
 alert('Vui lòng chọn ít nhất một yêu cầu!');  
 }  
});

**Giải thích**:

* $('.recordCheckbox:checked'): Chọn tất cả các checkbox đã được chọn trong bảng.
* .find('td:eq(1)').text(): Lấy giá trị từ cột chứa mã yêu cầu trong hàng tương ứng.
* selectedClaims: Là mảng lưu trữ danh sách các mã yêu cầu đã được chọn.

### ****Tổng kết****

* **Tạo bảng HTML**: Bảng HTML giúp hiển thị dữ liệu dưới dạng lưới, bao gồm các thẻ <thead>, <tbody>, <tr>, và <td>.
* **Đưa dữ liệu động vào bảng HTML**: Sử dụng JavaScript để lấy dữ liệu từ server và hiển thị vào bảng HTML thông qua vòng lặp và phương thức append.
* **Sử dụng checkbox trong bảng**: Checkbox giúp người dùng có thể chọn một hoặc nhiều hàng trong bảng.
* **Thêm nút chức năng**: Các nút chức năng như "Chấp nhận", "Từ chối" được sử dụng để điều khiển trạng thái của các yêu cầu hoặc thực hiện các hành động dựa trên lựa chọn của người dùng.

Thông qua các kỹ thuật trên, sinh viên có thể tạo ra các bảng quản lý danh sách dữ liệu động và tương tác với các phần tử trong bảng để thực hiện các thao tác như chọn hàng, gửi dữ liệu, và xử lý yêu cầu trên giao diện web.