**Lab01**

보고서

Lab01 문제해결 프로그래밍 설계 1분반

32207522 양상훈

제출일 : 25/03/18

**실행 결과**

스크린샷, 멀티미디어 소프트웨어, 지구, 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.텍스트, 소프트웨어, 멀티미디어 소프트웨어, 그래픽 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.텍스트, 소프트웨어, 멀티미디어 소프트웨어, 웹 페이지이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.스크린샷, 텍스트, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

**소스코드**

**index.html**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Lab01-32207522-양상훈</title>

<link rel="stylesheet" href="style.css">

</head>

<body>

<!-- 상단 이미지 + Natural Disaster Archive -->

<div class="title-container">

<img id="disaster-image" class="main-image" src="images/natural-disaster.jpg">

<h1 id="title" class="main-title">Natural Disaster Archive</h1>

</div>

<!-- 하단 컨테이너 부분 -->

<div class="container">

<div class="nav"> <!-- 재해 제목들을 나열 -->

<h3 id="volcano">Volcano</h3>

<h3 id="tornado">Tornado</h3>

<h3 id="typhoon">Typhoon</h3>

<h3 id="tsunami">Tsunami</h3>

</div>

<h3 class="table-title"></h3> <!-- 클릭된 재해 테이블의 이름 표시 -->

<table>

<thead class="thead"></thead> <!-- 테이블 헤더 -->

<tbody class="tbody"></tbody>

</table>

</div>

<!-- papaparse는 브라우저 환경에서 CSV 데이터를 처리하는데 유용하다.

브라우저에서 사용하기 위해서는 papaparse.min.js 파일을 <script>태그로 HTML에 포함시켜 사용한다.-->

<script src="https://cdnjs.cloudflare.com/ajax/libs/PapaParse/5.3.2/papaparse.min.js"></script>

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

</body>

</html>

**script.js**

// 각 자연재해에 대한 csv 파일의 경로를 저장한 객체

const csvFiles = {

volcano: "data/volcano.csv",

tornado: "data/tornado.csv",

typhoon: "data/typhoon.csv",

tsunami: "data/tsunami.csv"

};

// 재해 데이터를 저장할 객체

const disasterData = {};

// CSV 데이터를 비동기적으로 불러와서 처리하는 함수

function loadCSVData(callback) {

let filesLoaded = 0; // 로딩 완료된 파일의 개수

const totalFiles = Object.keys(csvFiles).length; // 총 파일 개수

Object.keys(csvFiles).forEach(disaster => {

fetch(csvFiles[disaster]) // csv 파일 가져오기

.then(response => response.text()) // 가져온 csv파일을 텍스트(문자열)로 변환

.then(csv => { // csv 파일을 papaParse를 이용하여 파싱하고 JS 객체 배열로 변환할 것임

Papa.parse(csv, {

header: true, // 첫 줄은 헤더(열 이름)로 이용

skipEmptyLines: true, // 빈 줄 무시

complete: function(results) {

disasterData[disaster] = results.data;

console.log(`${disaster} data loaded:`, disasterData[disaster]); // 데이터 확인

filesLoaded++;

// 모든 CSV 파일이 다 로딩된 후에 테이블을 그리도록 하기 위함.

if (filesLoaded === totalFiles && callback) {

callback(); // 모든 파일이 로드되었으면 callback 호출

}

}

});

})

.catch(err => console.error(`${disaster} CSV 로딩 실패:`, err));

});

}

// 페이지 로드 시 localStorage 확인하여 해당 disaster로 초기화

document.addEventListener("DOMContentLoaded", () => {

loadCSVData(() => {

// CSV 파일이 모두 로드 되면 실행할 것임.

const savedDisaster = localStorage.getItem("selectedDisaster");

if (savedDisaster && disasterData[savedDisaster]) {

makePage(savedDisaster);

document.getElementById(savedDisaster)?.classList.add('focus'); // focus 클래스를 추가함. 해당 ID 없으면 넘어감(?. : 옵셔널 체이닝)

}

});

// 자연재해 클릭 시 해당 화면으로 이동

document.getElementById('volcano')?.addEventListener('click', () => onLinkClick("volcano"));

document.getElementById('tornado')?.addEventListener('click', () => onLinkClick("tornado"));

document.getElementById('typhoon')?.addEventListener('click', () => onLinkClick("typhoon"));

document.getElementById('tsunami')?.addEventListener('click', () => onLinkClick("tsunami"));

// title 클릭시 처음 기본 화면

document.getElementById('title')?.addEventListener('click', () => onTitleClick());

});

function onLinkClick(disaster) {

['volcano', 'tornado', 'tsunami', 'typhoon'].forEach(id => {

document.getElementById(id)?.classList.toggle('focus', id === disaster);

});

localStorage.setItem("selectedDisaster", disaster);

makePage(disaster);

}

// Natural Disaster Archive를 클릭하면 초기 화면으로 이동함.

function onTitleClick() {

document.querySelector(".main-image").src = "images/natural-disaster.jpg";

document.querySelector(".table-title").textContent = "";

document.querySelector(".thead").innerHTML = ""; // 헤더 비움

document.querySelector(".tbody").innerHTML = ""; // 데이터 비움

localStorage.removeItem("selectedDisaster"); // 재해 선택한 스토리지를 비움(메인 화면으로 간 경우 이전 데이터 초기화 시킴)

// focus 클래스 제거

['volcano', 'tornado', 'tsunami', 'typhoon'].forEach(id => {

document.getElementById(id)?.classList.remove('focus');

});

}

function makePage(disaster) {

// document.getElementById('title').innerText = "Natural Disaster Archive"; 굳이 작성 안해도 됨

document.getElementById('disaster-image').src = `images/${disaster}.jpg`; // 이미지 설정

document.querySelector('.table-title').innerText = disaster.charAt(0).toUpperCase() + disaster.slice(1); // 테이블 이름 설정, 첫 글자 대문자 변환

const tbody = document.querySelector('.tbody');

tbody.innerHTML = "";

if (!disasterData[disaster]) return; // 데이터가 존재하지 않으면 종료

makeTable(disasterData[disaster], disaster);

}

function makeTable(arr, disaster) {

const tbody = document.querySelector('.tbody');

tbody.innerHTML = "";

const thead = document.querySelector('.thead');

thead.innerHTML = "";

// disaster에 따라 header가 다름

let headers = [];

if (disaster === 'tornado') {

headers = ["Year", "Month", "Day", "Date Time (UTC)", "Magnitude", "Injuries", "Fatalities", "Loss"];

} else if (disaster === 'tsunami') {

headers = ["ID", "Year", "Month", "Day", "Hour", "Latitude", "Longitude", "Location", "Country", "Cause", "Intensity", "Total Damage Description"];

} else if (disaster === 'typhoon') {

headers = ["ID", "Name", "Date", "Time", "Status", "Latitude", "Longitude", "Max Wind", "Min Pressure"];

} else if (disaster === 'volcano') {

headers = ["Year", "Volcano Name", "Location", "Country", "Lat / Long", "Type", "VEI", "Total Deaths"];

}

// tr 요소를 생성, 각 헤더 항목을 th로 만들어서 thead에 추가한다.

const trHeader = document.createElement('tr');

headers.forEach(header => {

const th = document.createElement('th');

th.innerText = header;

trHeader.appendChild(th);

});

thead.appendChild(trHeader);

// 각각의 데이터를 테이블에 추가한다.

// item 객체에서 해당 데이터를 가져와 td에 채워넣는다. 데이터가 없는 경우 'N/A' 로 표시한다.

arr.forEach(item => {

const tr = document.createElement('tr');

if (disaster === 'tornado') {

tr.innerHTML = `

<td>${item.yr || 'N/A'}</td>

<td>${item.mo || 'N/A'}</td>

<td>${item.dy || 'N/A'}</td>

<td>${item.datetime\_utc || 'N/A'}</td>

<td>${item.mag || 'N/A'}</td>

<td>${item.inj || 'N/A'}</td>

<td>${item.fat || 'N/A'}</td>

<td>${item.loss || 'N/A'}</td>

`;

} else if (disaster === 'tsunami') {

tr.innerHTML = `

<td>${item.ID || 'N/A'}</td>

<td>${item.YEAR || 'N/A'}</td>

<td>${item.MONTH || 'N/A'}</td>

<td>${item.DAY || 'N/A'}</td>

<td>${item.HOUR || 'N/A'}</td>

<td>${item.LATITUDE || 'N/A'}</td>

<td>${item.LONGITUDE || 'N/A'}</td>

<td>${item.LOCATION\_NAME || 'N/A'}</td>

<td>${item.COUNTRY || 'N/A'}</td>

<td>${item.CAUSE || 'N/A'}</td>

<td>${item.TS\_INTENSITY || 'N/A'}</td>

<td>${item.DAMAGE\_TOTAL\_DESCRIPTION || 'N/A'}</td>

`;

} else if (disaster === 'typhoon') {

tr.innerHTML = `

<td>${item.ID || 'N/A'}</td>

<td>${item.Name || 'N/A'}</td>

<td>${item.Date || 'N/A'}</td>

<td>${item.Time || 'N/A'}</td>

<td>${item.Status || 'N/A'}</td>

<td>${item.Latitude || 'N/A'}</td>

<td>${item.Longitude || 'N/A'}</td>

<td>${item['Maximum Wind'] || 'N/A'}</td>

<td>${item['Minimum Pressure'] || 'N/A'}</td>

`;

} else if (disaster === 'volcano') {

tr.innerHTML = `

<td>${item.Year || 'N/A'}</td>

<td>${item.Name || 'N/A'}</td>

<td>${item.Location || 'N/A'}</td>

<td>${item.Country || 'N/A'}</td>

<td>${item.Latitude && item.Longitude ? `${item.Latitude} / ${item.Longitude}` : 'N/A'}</td>

<td>${item.Type || 'N/A'}</td>

<td>${item.VEI || 'N/A'}</td>

<td>${item.Deaths || 'N/A'}</td>

`;

}

tbody.appendChild(tr);

});

}

**style.css**

.container{

display: flex;

flex-direction: column; /\* 위 -> 아래 정렬 \*/

justify-content: center; /\* flex-direction: column 이므로 세로 방향 + 가운데 정렬 \*/

align-items: center; /\* flex-direction: column 이므로 가로 방향 + 가운데 정렬 \*/

}

.nav{

display: flex;

flex-direction: row;

width:50%; /\* 부모 요소, 혹은 화면 전체의 너비의 절반\*/

justify-content: space-between; /\* 가로 방향 + 여백으로 요소들 나열 \*/

}

.main-image{

width: 100%; /\* 부모(container)에 맞춰 꽉 채움 \*/

max-width: 600px; /\* 최대 너비 제한 \*/

height: auto; /\* 가로 크기에 맞춰 비율 자동 조정 \*/

display: block; /\* 이미지가 줄 하나 다 차지하게 만들기 \*/

margin: 0 auto; /\* 이미지가 가운데로 정렬됨. display: block과 세트로 자주 씀 \*/

}

.main-title{

font-size: 2em;

margin-top: 10px;

margin-bottom: 0;

text-align: center;

}

h1{

color: blue;

}

h3{

cursor: pointer;

}

.focus{

color: green;

}

table{

border-collapse: collapse;

text-align: left;

line-height: 1.5;

border-top: 1px solid #ccc;

border-left: 1px solid #ccc;

margin : 20px 10px;

}

th{

background-color: #f2f2f2; /\* 밝은 회색 배경 \*/

width: 147px;

padding: 10px;

font-weight: bold;

vertical-align: top;

color: #153d73;

border-right: 1px solid #ccc;

border-bottom: 1px solid #ccc;

text-align: center;

}

td{

width: 349px;

padding: 10px;

vertical-align: top;

border-right: 1px solid #ccc;

border-bottom: 1px solid #ccc;

text-align: center;

}

.hidden{

display: none;

}

**코드 설명, 기능 구현**

**Index.html**

papaparse 외에도 여러 파싱 라이브러리가 있는데 왜 papaparse를 사용했는지?

브라우저 환경에서 CSV파일을 처리하는데 유용하기 때문이다.

브라우저에서 사용하기 위해서는 papaparse.min.js 파일을 <script>태그로 HTML에 포함시켜 사용한다.

**style.css**

vitamins & minerals에서

main-image, main-title을 추가하였다.

main-image는 jpg 이미지의 크기를 조정한다.

main-title은 이미지 아래 글의 위아래 여백 조절, 가운데 정렬한다.

th에 회색을 입혔다. th,td 모두 가운데 정렬시켰다.

**script.js**

csvFiles : 내가 사용할 자연재해 이름을 설정하고 그에 맞는 csv파일의 경로를 설정하였다.

disasterData: 파싱한 데이터를 저장할 것이다.

loadCSVdata(callback) : csv 파일을 불러와서 문자열로 변환하고 papaparse를 이용하여 파싱한 데이터를 disasterData에 저장한다. 모든 csv파일이 로딩이 된 후에 테이블을 그리기 위해서 call, callback()을 사용한다.

document.addEventListener : 내가 만약 volcano를 클릭한 상태에서 화면을 나갔다 다시 들어오면 save된 정보, saveDisaster에 관한 페이지가 그대로 나오도록 구현

onTitleClick : main-title을 클릭하는 경우 메인 화면으로 돌아감.

onLinkClick : disaster 종류를 클릭하는 경우 그에 맞는 화면을 생성함.

makePage : disaster에 맞는 이미지, title을 설정하고 table을 만들기 시작함.

makeTable: makePage에서 호출된다. Csv 파일마다 헤더가 다르다. 내가 보여주고 싶은 데이터만 작성을 한다. 빈칸인 경우 N/A로 설정하였다.