최종 보고서

웹프로그래밍

학 과	산업보안학과
학 번	32233421, 32232146)
이 름	이은지, 서재린
제 출 일	2024-06-17

내용

제출 전에 목차 전체 업데이트 필수

1. 기능 시나리오

각 기능 시나리오 별로 사용법을 **개괄식**으로 작성
사용한 Open API 반드시 명시

기능 1. 영화관 위치 제공 -----

사용한 API: : https://apis.map.kakao.com/web/ (카카오맵 API)

분당에 있는 모든 영화관(CGV)의 위치를 제공하여 사용자에게 역 근처의 영화관 정보를 제공한다. 역이름만으로도 가장 가까운 영화관을 카카오 맵을 통해 마커로 표시한다.

-카카오맵 api 를 사용하여 지도를 불러오고, 이용자로 하여금 역 근처의 영화관을 찾을 수 있도록 도움을 준다

기능 2. 영화 상세정보 제공 -------

사용한 API: https://www.kobis.or.kr/kobisopenapi/homepg/apiservice/searchServiceInfo.do

(주간/ 주말 박스오피스 정보)

주간/주말의 영화 순위를 나타내고 각 영화별로 영화코드를 제공한다. 영화코드를 입력 시 영화에 대한 상세정보(영화제목, 상영시간, 영화 유형, 개봉일)을 제공한다

- kobis 의 주간/ 주말 박스오피스 순위 api 를 활용하여 고유의 영화코드를 제공받고, 이 후 박스오피스 정보 api 를 통해 영화 관련 정보를 제공한다.

기능 3. 예술의 전당 종합 공연 정보 제공 -----

사용한API:https://www.culture.go.kr/data/openapi/openapiView.do?id=570&category=A&gubun=A

(예술의 전당 종합 공연 정보(전체 장르))

장르 확인 버튼을 눌러 장르 리스트를 확인하고, 하나의 장르를 골라 장르 입력 칸에 입력한 후 확인 버튼을 누른다. 예를 들어 뮤지컬을 골랐다고 하면, 뮤지컬 제목과 날짜, 상세 정보를 네 개 보여준다. 마음에 드는 공연의 복사하기 버튼을 누르면 공연의 제목과 날짜, 상세 정보가 클립 보드에 복사된다.

기능 4. 키워드로 도서 검색-----

사용한 API: https://developers.kakao.com/docs/latest/ko/daum-search/dev-guide (다음 책 서비스, 도서 정보 검색)

키워드 입력 칸에 검색하고 싶은 키워드를 입력한 후 검색 버튼을 누르면 그 키워드가 포함된 도서의

표지, 제목, 저자, 상세정보를 보여준다. 책 표지를 클릭하면 다음 책 서비스의 책 소개 페이지로 넘어가 더 자세한 정보를 알 수 있다.

2. 최소 개발 구현

최소 개발 기준을 어떻게 만족하였는지 **개괄식**으로 작성
해당하는 소스코드 및 line number를 반드시 포함.

■ 서로 다른 <input> 3개 이상 활용

영화관 위치.html 17번째 줄

```
<input type="text" id="station-input" placeholder="역 이름을 입력하세요"> <!-- <input> (1) -->
```

박스오피스.html 48번째 줄

```
<input type="text" id="movie-code" placeholder="영화 코드를 입력하세요">  <!-- <input> (2)-->
```

공연정보.html의 36th line

```
<input type="text" class="input" id="genreInput" value="장르 입력">
```

도서정보.html의 27th line

```
<input type="text" class="input" id="keyword" placeholder="키워드 입력">
```

■ List 및 table 필수 활용

박스오피스.html 22번째 줄

```
<!-- 영화 정보를 표시할 <table> 사용 -->
<div id="movie-info">
    <h2>영화 정보</h2>
    <table id="movie-table">
        <thead>
            <tr>
                <th>순위</th>
                <th>영화 제목</th>
                <th>영화 코드</th>
            </tr>
        </thead>
        <tbody id="movie-list"></tbody>
    </table>
</div>
```

박스오피스.html 37번째 줄

```
<!-- 영화 상세 정보를 표시할 <table> 사용 -->
<div id="movie-detail" style="display: none;">
    <h2>영화 상세 정보</h2>
    <table id="movie-detail-table">
        <tbody id="movie-detail-info"></tbody>
    </table>
</div>
```

공연정보.html의 26th line

```
<ul class="genreList" id="genreList">
    <li>오페라</li>
    <li>연극</li>
    <li>무용</li>
    <li>뮤지컬</li>
    <li>콘서트</li>
    <li>재즈</li>
    <li>크로스오버</li>
    <li>복합장르</li>
</ul>
```

■ Semantic (header, navigation, footer) 필수 활용

페이지 /

웹프 과제.html 11, 17, 39번째 줄

```
<header> <!--Semantic (header, navigation, footer)-->
<nav> <!--Semantic (header, navigation, footer)-->
<nav> <!--Semantic (header, navigation, footer)-->
<footer> <!--Semantic (header, navigation, footer)-->
```

영화관 위치.html 11, 15, 118번째 줄

```
<header> <!--Semantic (header, navigation, footer)-->
<nav class="clearfix"> <!--Semantic (header, navigation, footer)-->
<footer> <!--Semantic (header, navigation, footer)-->
```

박스오피스html 11, 46, 163번째 줄

```
<header> <!--Semantic (header, navigation, footer)-->
<nav> <!--Semantic (header, navigation, footer)-->
<footer> <!--Semantic (header, navigation, footer)-->
```

공연정보.html의 11th, 19th, 68th line

```
<header> <!-- semantic(header, navigation, footer) 활용-->
    <div class="title">
        <h1>예술의 전당 공연 정보 확인, 공유</h1>
    </div>
</header>
```

```
<nav> <!-- semantic(header, navigation, footer) 활용-->
    <ul>
        <a href="웹프 과제.html" title="go-main">메인 화면으로 돌아가기</a></li>
    </ul>
</nav>
```

```
<footer> <!-- semantic(header, navigation, footer) 활용-->
    <div class="footer-content">
        <p>2024 web programming project</p>
    </div>
</footer>
```

도서정보.html의 12th, 22th, 39th line

```
<header> <!-- semantic(header, navigation, footer) 활용-->
    <div class="title">
        <h1>키워드로 도서 검색</h1>
    </div>
</header>
```

```html
<nav> <!-- semantic(header, navigation, footer) 활용-->
    <ul>
        <a href="웹프 과제.html" title="go-main">메인 화면으로 돌아가기</a></li>
    </ul>
</nav>
```

```html
<footer> <!-- semantic(header, navigation, footer) 활용-->
    <div class="footer-content">
        <p>2024 web programming project</p>
    </div>
</footer>
```

■Pseudo-class 2개 이상

style.css 69번째 줄

```css
/* Pseudo-class를 사용하여 버튼에 호버 효과(색깔 변환) 추가 */
.button:hover {
    background-color: #8a9ee8;
}
```

style2.css 59번째 줄

```css
/* Pseudo-class를 사용하여 버튼에 호버 효과(색깔 변환) 추가 */
button:hover {
    background-color: #8a9ee8;
}
```

공연정보.CSS의 39th, 111th line

```css
button:hover {  /* Pseudo-class 활용
    background-color: #8a9ee8;
}
```

```css
.card:hover {
    transform: scale(1.05);
}
```

도서정보.CSS의 42th line

```css
button:hover { /* Pseudo-class 활용 */
    background-color: #8a9ee8;
}
```

■ Pseudo-element 2개 이상

style2.css 75번째 줄

```
/* Pseudo-element를 사용하여
#map::before {
.card .card-detail { /
    max-height: 100px;
    overflow-y: auto;
}
}
```

```
footer p{ /*combinator 활용*/
    margin-top: auto;
    background-color: #ebddea;
    color: #333;
    padding: 20px;
    text-align: center;
    clear: both; /*clear 활용*/
    display: grid;
    justify-content: center;
    align-items: center;
}
```

도서정보.CSS의 85th line

```
.card::before { /* Pseudo-element 활용*/
    content: '';
    display: block;
    width: 100%;
    height: 10px;
    background-color: #ccc;
    margin-bottom: 10px;
}
```

■ attribute selector 2개 이상

style2.css 64번째 줄

```
/* attribute selector를 사용하여 특정 input에 스타일 적용 */
input[type="text"] {
    padding: 10px;
    margin: 10px 0;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 16px;
    width: 100%;
    max-width: 600px;
}
```

도서정보.CSS의 46th line

```
.button[disabled]
    opacity: 0.5;
}
```

■ combinator 5개 이상 활용

공연정보.CSS의 94th, 117th line

도서정보.CSS의 30th, 109th line

```
.button > nav {
    display: inline-block;
}
```

```
footer p{ /*combinator 활용*/
    margin-top: auto;
    background-color: #ebddea;
    color: #333;
    padding: 20px;
    text-align: center;
    clear: both; /*clear 활용*/
    display: grid;
    justify-content: center;
    align-items: center;
}
```

style2.css 28번째 줄

```
input, button {
```

style3.css 29번째 줄

```
input, button {
```

■Float 및 clear 활용하여 웹의 layout 구성 필수

style.css 65,74번째 줄

```
.buttons .button {
    float: left;
}
```

```
.clearfix::after {
    content: "";
    display: table;
    clear: both;
}
```

공연정보.CSS의 123th line

```
clear: both; /*clear 활용*/
```

도서정보.CSS의 115th line

```
clear: both; /*clear 활용*/
```

■잘못된 동작에 대한 에러 핸들링 두개 이상 구현 필수

박스오피스.html 101번째 줄

```
// 입력된 영화 코드가 숫자인지 확인 (에러 핸들링)
if (!(/^\d+$/.test(movieCode))) {
    alert('잘못된 영화 코드입니다. 숫자만 입력하세요.');
    return; // 유효하지 않은 영화 코드이므로 함수 종료
}
```

박스오피스.html 91번째 줄

```
.catch(error => {
    // 데이터를 불러오는 중 에러 발생 시 콘솔에 에러를 출력하고 사용자에게 알림(에러 핸들링)
    console.error('Error:', error);
    alert('데이터를 불러오는 중 오류가 발생했습니다.');
});
```

공연정보.js의 48th line

```
if (!genre) {  //잘못된 동작에 대한 에러 핸들링
    alert('유효하지 않은 장르입니다.');
    return;
}
```

도서정보.js의 7th line

```
if (!keyword) { // 잘못된 동작에 대한 에러 핸들링
    alert('키워드를 입력하세요');
    return;
}
```

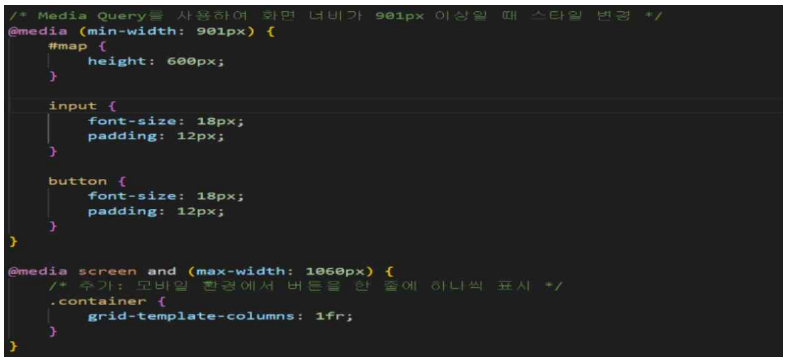
■모바일 환경 대응 구현 필수(width:900px 이하일때 동작, 기존 웹사이트는 width:1060px로 구현)

style.css 80, 51번째 줄

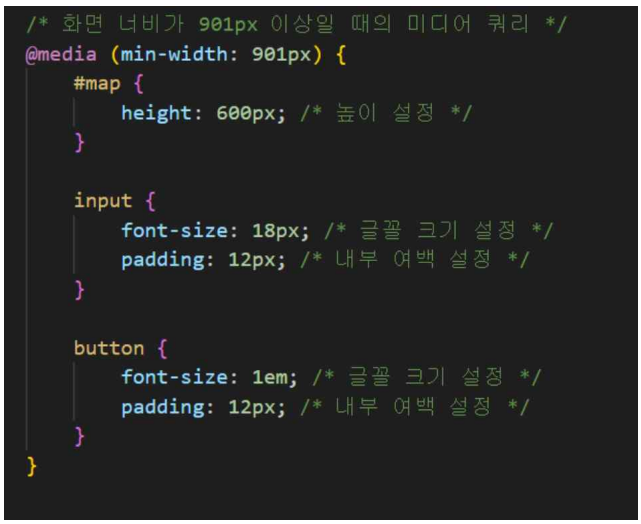
```
/* 태블릿 및 모바일 환경 */
@media screen and (max-width: 1060px) {
    .button {
        width: calc(100% - 20px); /* 모바일 환경에서는 버튼을 한 줄에 하나씩 표시 */
    }
}
```

```
/* 반응형 스타일 */
@media screen and (min-width: 901px) {
    .container {
        width: 1000px; /* 헤더 및 버튼 크기에 맞게 조정 */
    }
    .button {
        width: calc(50% - 20px);
    }
}
```

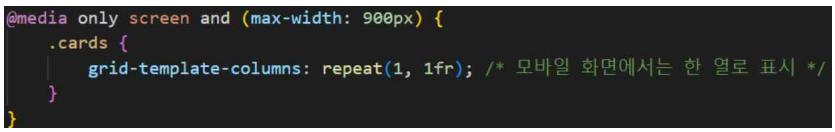
style2.css 83번째 줄



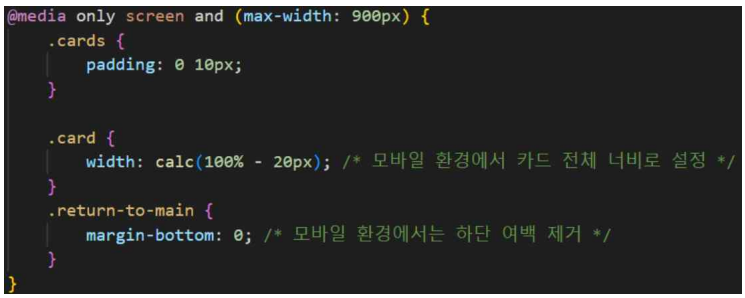
style3.css 71



공연정보.CSS의 130th line



도서정보.CSS의 121th line



■flex 혹은 grid 한번 이상 필수 활용

style2.css 20번째 줄

```css
/*container 클래스를 grid 컨테이너로 설정하고, 그리드 아이템들을 반응형으로 정렬*/
.container {
    margin: 20px auto;
    display: grid; /* grid 사용 */
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr)); /*  반응형 그리드 설정 */
    gap: 10px; /*그리드 아이템 사이의 간격 설정 */
}
```

공연정보.CSS의 4th, 20th, 64th, 82th line

```
display: flex;
```

```
display: grid; /* grid 활용 */
```

```
display: grid;
```

```
display: flex; /* flex 활용 */
```

도서정보.CSS의 4th , 19th , 57th , 74th line

```
display: flex;
```

```
display: grid;
```

```
display: grid;
```

```
display: flex;
```

■DOM 활용하여 HTML element 혹은 CSS property 합쳐서 6개 이상 다루기

공연정보.js의 3th, 21th, 45th, 68th, 72~74th line등

```
var genreInput = document.getElementById("genreInput");
```

```
const genreList = document.getElementById("genreList");
```

```
var genreInput = document.getElementById("genreInput").value;
```

```
var items = xmlDoc.getElementsByTagName('item');
```

```
document.querySelector(`.card-title${i + 1}`).textContent = item.getElementsByTagName('title')[0].textContent;
document.querySelector(`.card-date${i + 1}`).textContent = item.getElementsByTagName('regDate')[0].textContent.substring(0, 10);
document.querySelector(`.card-detail${i + 1}`).textContent = item.getElementsByTagName('description')[0].textContent;
```

■입력 Format, 네트워크 연결 등의 에러 처리 기능 필수 구현

영화관 위치.html 95

```
// 네트워크 연결 에러 처리
function fetchMovieData(url) {
    fetch(url)
        .then(response => {
            if (!response.ok) {
                throw Error(response.statusText);
            }
            return response.json();
        })
        .then(data => {
            // 성공적으로 데이터를 받아온 경우 처리하는 코드
        })
        .catch(error => {
            // 네트워크 연결 에러 처리
            console.error('네트워크 연결 에러:', error);
            alert('네트워크 연결에 문제가 있습니다.');
        });
}
```

박스오피스.html 140번째 줄

```
    // 네트워크 연결 에러 처리
    function fetchMovieData(url) {
        fetch(url)
            .then(response => {
                if (!response.ok) {
                    throw Error(response.statusText);
                }
                return response.json();
            })
            .then(data => {
                // 성공적으로 데이터를 받아온 경우 처리하는 코드
            })
            .catch(error => {
                // 네트워크 연결 에러 처리
                console.error('네트워크 연결 에러:', error);
                alert('네트워크 연결에 문제가 있습니다.');
            });
    }
```

공연정보.js 59th~96th line

```
    try {
        const response = await fetchWithTimeout(url + queryParams, { timeout: 5000 }); // 5초 내에 응답이 없으면 타임아웃 처리
        if (!response.ok) {
            throw new Error('네트워크 오류: ' + response.status);
        }

        const data = await response.text();
        var parser = new DOMParser();
        var xmlDoc = parser.parseFromString(data, 'text/xml');
        var items = xmlDoc.getElementsByTagName('item');

        for (let i = 0; i < Math.min(items.length, 4); i++) {
            const item = items[i];
            document.querySelector(`.card-title${i + 1}`).textContent = item.getElementsByTagName('title')[0].textContent;
            document.querySelector(`.card-date${i + 1}`).textContent = item.getElementsByTagName('regDate')[0].textContent.substring(0, 10)
            document.querySelector(`.card-detail${i + 1}`).textContent = item.getElementsByTagName('description')[0].textContent;
        }
    } catch (error) {
        alert('오류 발생: ' + error.message); // 네트워크 연결 에러 처리
    }
}
```

```
// 네트워크 요청이 일정시간 내 없을 경우 타임아웃처리
async function fetchWithTimeout(resource, options) {
    const { timeout = 8000 } = options;

    const controller = new AbortController();
    const id = setTimeout(() => controller.abort(), timeout);
    const response = await fetch(resource, {
        ...options,
        signal: controller.signal
    });
    clearTimeout(id);

    return response;
}
```

도서정보.js의 22th~28th, 29th~32th line

```
success: function(response) {
    if (response.documents.length > 0) {
        displayResults(response.documents[0]);
    } else {
        alert('검색 결과가 없습니다.'); // 검색 결과가 없는 경우 에러 처리
    }
},
error: function(xhr, status, error) {  //네트워크 에러 처리
    console.error('Error fetching data:', error);
    alert('네트워크 오류가 발생했습니다. 잠시 후 다시 시도해주세요.');
}
```

■수업시간에 배운 jQuery/AJAX/JSON 기능 활용

도서정보.js의 6th, 12th, 39th, 45~49th line

```
const keyword = $('#keyword').val();
$.ajax({    const card = $('.card');
img.attr('src', book.thumbnail);
title.text(book.title);
```

3. 추가 구현 [가산점]

최소 개발 기준 이외에 추가 구현한 부분을 **개괄식**으로 작성
해당하는 소스코드 및 line number를 반드시 포함.

■ 박스오피스.html

=> '메인화면으로 돌아가기' 기능 구현, 박스오피스.html 50번째 줄

```
        <button onclick="goToMainPage()">메인 화면으로 돌아가기</button>
    </div>
```

=> 에러 발생에 대한 경고창 제공, 박스오피스.html 130번째 줄

```
.catch(error => {
    // 에러가 발생하면 콘솔에 에러를 출력하고 사용자에게 알림
    console.error('Error:', error);
    alert('영화 정보를 불러오는 중 오류가 발생했습니다.');
});
```

■영화관 위치.html

=> '메인화면으로 돌아가기' 기능 구현, 영화관 위치.html 89번째 줄

```
// 메인 페이지로 돌아가기 함수
function goToMainPage() {
    window.location.href = "웹프 과제.html"; // 메인 페이지 경로로 이동
}
```

=> 마커 클릭 이벤트, 영화관 위치.html 78번째 줄

```
// 마커 클릭 이벤트 추가
kakao.maps.event.addListener(marker, 'click', function() {
    infowindow.open(map, marker);
});
```

=> 마커 클릭시 인포위도우 설정, 영화관 위치.html 73번째 줄

```
// 마커 클릭 시 표시할 인포윈도우 내용 설정
var infowindow = new kakao.maps.InfoWindow({
    content: stationName + " 인근 CGV"
});
```

=> 초기 지도 중심을 위도, 경도 설정을 통해 단국대로 설정, 영화관 위치.html 30번째 줄

```
// 지도 표시할 div와 초기 옵션 설정
var mapContainer = document.getElementById('map'),
    mapOption = {
        center: new kakao.maps.LatLng(37.32197224820124, 127.12519889261033), // 초기 중심 좌표
        level: 3 // 지도의 초기 확대 레벨
    };
```

=> 지도 중심이동, 영화관 위치.html 83번째 줄

```
    // 지도의 중심을 해당 위치로 이동
    map.setCenter(coords);
    // 지도의 확대 레벨 조정
    map.setLevel(4);
}
```

■ style.css, style2.css, style3.css

=> 전체적인 호버효과, style.css 70번째 줄, style2.css 60번째 줄, style3.css 52번째 줄

```
button:hover {
    background-color: #8a9ee8;
```

■ 공연정보.html

button의 onclick event를 활용하여 js 함수를 불러오도록 함(25th, 37th, 45th, 52th, 57th, 63th line)

```
<button class="button" onclick="genreButton()">장르 확인</button>
<button class="button" onclick="fetchPerformances()">확인</button>
<button class="share-button share-button1" onclick="copyContent()">복사하기</button>
```

■공연정보.css

디자인을 위해 border-radius 사용(15th, 67th)

```
border-radius: 10px;
```

justify-content property활용(21th)

```
justify-content: center;
```

마우스 커서를 포인터로 변경하여 클릭 가능함을 표시 (36th)

```
cursor: pointer;
```

pseudo-class(hover)사용해 버튼에 커서 올리면 버튼 색깔 변환(39th)

```
button:hover {  /* Pseudo-class
    background-color: #8a9ee8;
}
```

display: none 사용해 장르 리스트 초기 표시 설정을 숨김으로 함(52th),

```
.genreList {
    display: none;
    list-style-type: none;
    padding: 0;
    margin: 0 20px;
}
```

display: block으로 장르리스트를 블록 레벨로 표시해 숨겨졌던 장르리스트가 보이게 됨(59th)

```
.show {
    display: block;
}
```

```
genreList.classList.toggle("show");
```
(공연정보.js – 22th)

grid-template-columns: repeat(2, 1fr);으로 카드가 2개의 열이고, 각 열 동일한 너비 갖도록 함(65th)

```
63   .cards {
64       display: grid; /*grid 활용 */
65       grid-template-columns: repeat(2, 1fr);
```

카드에 옅은 그림자를 생성해 돋보이도록 함(77th)

```
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

calc()함수 사용해 .cards의 너비에 따라 카드 너비 동적 계산, 카드 양쪽에 20픽셀식 여백 확보(78th~80th)

```
padding: 20px;
text-align: center;
width: calc(100% - 40px);
```

overflow-y: auto로 카드 디테일이 카드의 수직 크기 보다 넘치면 스크롤 가능하도록 설정(96th)

```
overflow-y: auto;
```

커서 올리면 카드 1.05배 커짐(hover이용) (112th)

```
.card:hover {
    transform: scale(1.05);
}
```

■공연정보.js

async, await 이용해 공연 정보를 비동기적으로 가져옴(44th ~79th)

```
async function fetchPerformances() {
```

```
const response = await fetchWithTimeout(
```

try/catch 사용, DOMParser.parseFromString()사용(59th~78th)

```
try {
    const response = await fetchWithTimeout(url + queryParams, { timeout: 5000 }); // 5초 내에 응답이 없으면 타임아웃 처리
    if (!response.ok) {
        throw new Error('네트워크 오류: ' + response.status);
    }

    const data = await response.text();
    var parser = new DOMParser();
    var xmlDoc = parser.parseFromString(data, 'text/xml');
    var items = xmlDoc.getElementsByTagName('item');

    for (let i = 0; i < Math.min(items.length, 4); i++) {
        const item = items[i];
        document.querySelector(`.card-title${i + 1}`).textContent = item.getElementsByTagName('title')[0].textContent;
        document.querySelector(`.card-date${i + 1}`).textContent = item.getElementsByTagName('regDate')[0].textContent.substring(0, 10
        document.querySelector(`.card-detail${i + 1}`).textContent = item.getElementsByTagName('description')[0].textContent;
    }
} catch (error) {
    alert('오류 발생: ' + error.message); // 네트워크 연결 에러 처리
}
}
```

Abortcontroller()로 요청이 일정시간 내 없을 경우 작업 중간에 취소 (87th)

```
async function fetchWithTimeout(resource, options) {
    const { timeout = 8000 } = options;

    const controller = new AbortController();
    const id = setTimeout(() => controller.abort(), timeout);
    const response = await fetch(resource, {
        ...options,
        signal: controller.signal
    });
    clearTimeout(id);

    return response;
}
```

createElement()로 element 생성, appendChild()로 element를 document에 추가, select()로 element 선택 후 document.execCommand('copy') 호출해 텍스트를 클립 보드에 복사,마지막으로 textarea element를 document에서 제거(103th~ 118th)

```
function copyContent() {
    var title = document.querySelector('.card-title1').innerText;
    var date = document.querySelector('.card-date1').innerText;
    var detail = document.querySelector('.card-detail1').innerText;

    var copyText = title + '\n' + date + '\n' + detail;

    var textarea = document.createElement('textarea');
    textarea.value = copyText;
    document.body.appendChild(textarea);
    textarea.select();
    document.execCommand('copy');
    document.body.removeChild(textarea);

    alert('내용이 복사되었습니다!');
}
```

■ 도서정보.html

a tag의 title attribute 활용(24th)

```
<a href="웹프 과제.html" title="go-main">메인 화면으로 돌아가기</a>
```

button의 onclick event를 활용하여 js 함수를 불러오도록 함(28th)

```
<button class="button" onclick="searchBooks()">검색</button>
```

■ 도서정보.css

border-radius로 테두리 둥글게(15th, 38th, 52th)

```
border-radius: 10px;
```

마우스 커서를 포인터로 변경하여 클릭 가능함을 표시(39th)

```
cursor: pointer;
```

button클래스중 disabled속성 반투명하게 지정(47th)

```
.button[disabled] {
    opacity: 0.5;
}
```

card에 옅은 그림자 생성

```
box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
```

pseudo-element의 ::before사용해 .card의 시작 부분에 선 추가해 디자인 추가(85th~ 92th)

```
.card::before { /* Pseudo-elem
    content: '';
    display: block;
    width: 100%;
    height: 10px;
    background-color: #ccc;
    margin-bottom: 10px;
}
```

overflow-y로 카드 디테일이 카드의 수직 크기를 넘어설 때 수직 스크롤 추가(95th)

```
overflow-y: auto;
```

■ 도서정보. js

_black로 새 탭 열어 이동(52th)

```
card.click(function() {
    window.open(book.url, '_blank');
});
```