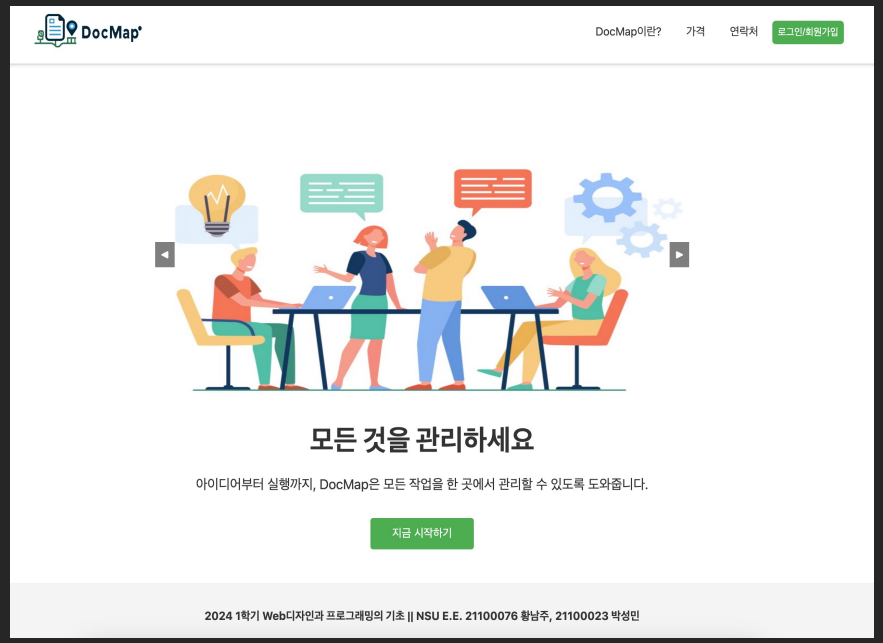
Doc  Map

문서를 쉽게 “DocMap” 하자!

21100023 박 성 민

21100076 황 남 주

배경설명



저희는 대학 생활에서의 과제와 프로젝트를 효율적으로 관리하기 위한 도구를 개발했습니다.

이를 통해 강의 일정부터 과제, 그리고 팀 프로젝트까지 모두 체계적으로 관리할 수 있습니다.

학생들이 스트레스 없이 학업을 조직하고 효과적으로 협업할 수 있는 서비스를 제공하고 싶었습니다.

HTML

```html
<!DOCTYPE html>
<html lang="ko">

<head>
    <meta chars
    <meta name=
    <!-- 외부 
    <link rel="
    <!-- jQuery
    <script src
        integri
    <!-- FontAw
    <script src
    <!-- 외부 
    <script src

    <title>Docl
</head>
```

```html
<body>
    <header>
        <nav>
            <div class="logo">
                <a href="/index.html">

                </a
            </div>
            <div>
                <a 
                <a 
                <a 
                <spa
                <but
                <but
            </div>
        </nav>
    </header>
    <div id="loginM
        <div class="
            <span c
            <div cl
                <di
                </d
            </div>
```

```html
<section class="sub-section" id="sub-section">
        <div class="nav-links">
            <!-- 디자인 섹션으로 이동하는 네비게이션 버튼 -->
            <button class="nav-button" data-target="design">
                <i class="fa-solid fa-palette"></i>
                <span>디자인</span>
            </button>
            <!-- 문서 섹션으로 이동하는 네비게이션 버튼 -->
            <button class="nav-button" data-target="docs">
                <i class="fa-solid fa-file"></i>
                <span>문서</span>
            </button>
            <!-- 메모 섹션으로 이동하는 네비게이션 버튼 -->
            <button class="nav-button" data-target="memo">
                <i class="fa-solid fa-file-invoice"></i>
                <span>메모</span>
            </button>
            <!-- 캘린더 섹션으로 이동하는 네비게이션 버튼 -->
            <button class="nav-button" data-target="calendar">
                <i class="fa-solid fa-calendar-days"></i>
                <span>캘린더</span>
            </button>
            <!-- 프로젝트 섹션으로 이동하는 네비게이션 버튼 -->
            <button class="nav-button" data-target="project">
                <i class="fa-solid fa-diagram-project"></i>
                <span>프로젝트</span>
            </button>
        </div>
        <!-- 디자인 섹션 내용 -->
        <div class="content-container">
            <div class="content" id="design">
                <p>DocMap으로 창의적인 아이디어로 디자인을 형상화하세요.</p>
                <img src="https://cdn.pixabay.com/photo/2018/08/18/13/26/interface-3614766_1280.png" alt="Design">
            </div>
        </div>
```

CSS

```css
@import url(http://fonts.googleapis.com/earlyaccess/notosanskr.css); /* 구글 폰트 불러오기 */

body {
    font-family: 'Noto Sans KR', -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif; /* 폰트 설정 */
    margin: 0; /* 기본 마진 제거 */
    padding: 0; /* 기본 패딩 제거 */
    background-color: #fff; /* 배
    color: #333; /* 글자색 진회색
}

header {
    background-color: #fff; /* 배
    border-bottom: 1px solid #cc
    box-shadow: 0 2px 4px rgba(0
    padding: 1rem; /* 내부 여백
    position: fixed; /* 고정 위치
    width: 100%; /* 너비 100% 설
    top: 0; /* 상단에 위치 */
    left: 0; /* 왼쪽에 위치 */
    z-index: 1000; /* z-index 설
}

header nav {
    display: flex; /* 플렉스 박스
    justify-content: space-betwe
    align-items: center; /* 중앙
    max-width: 1200px; /* 최대 너
    margin: 0 auto; /* 가운데 정
}

header nav a {
    margin: 0 1rem; /* 좌우 마진
    text-decoration: none; /* 밑
    color: #333; /* 글자색 설정
}
```

```css
#loginForm, #signupForm {
    display: flex; /* 플렉스 박스 사
    flex-direction: column; /* 세로
}

#loginForm label, #signupForm label
    margin-bottom: 5px; /* 아래쪽 마
    color: #333; /* 글자색 설정 */
}

#loginForm input, #signupForm input
    margin-bottom: 15px; /* 아래쪽
    padding: 10px; /* 내부 여백 설정
    border: 1px solid #ddd; /* 테두
    border-radius: 5px; /* 테두리 둥
    font-size: 16px; /* 글자 크기 설
}

#loginForm .button-container {
    display: flex; /* 플렉스 박스 사
    justify-content: space-between;
    gap: 10px; /* 간격 설정 */
}

#loginForm button, #signupButton, #
    flex: 1; /* 플렉스 아이템 1 설정
    margin-top: 10px; /* 위쪽 마진
    padding: 10px; /* 내부 여백 설정
    color: white; /* 글자색 흰색 설
    background-color: #4caf50; /* 배
    border: none; /* 테두리 제거 */
    border-radius: 5px; /* 테두리 둥
    font-size: 16px; /* 글자 크기 설
    cursor: pointer; /* 커서 포인터
    box-sizing: border-box; /* 박스
}
```

```css
.gallery img {
    width: 800px; /* 이미지 너비 설정 */
    height: 400px; /* 이미지 높이 설정 */
    display: block; /* 블록 요소 설정 */
}

.arrow {
    position: absolute; /* 절대 위치 설정 */
    top: 50%; /* 상단에서 50% 위치 */
    transform: translateY(-50%); /* 수직 중앙 정렬 */
    background-color: rgba(0, 0, 0, 0.5); /* 배경색 반투명 설정 */
    color: white; /* 글자색 흰색 설정 */
    padding: 0.5rem; /* 내부 여백 설정 */
    cursor: pointer; /* 커서 포인터 설정 */
    z-index: 100; /* z-index 설정 */
}

.left-arrow {
    left: 10px; /* 왼쪽에서 10px 위치 */
}

.right-arrow {
    right: 10px; /* 오른쪽에서 10px 위치 */
}

.main-section h1 {
    font-size: 2.5rem; /* 글자 크기 설정 */
    margin-bottom: 1rem; /* 아래쪽 마진 설정 */
}

.main-section p {
    font-size: 1.2rem; /* 글자 크기 설정 */
    line-height: 1.6; /* 줄 간격 설정 */
    margin-bottom: 2rem; /* 아래쪽 마진 설정 */
}
```

부드러운 스크롤 기능 / 로그인

```
5
6   // 네비게이션 링크 클릭 시 부드러운 스크롤 기능 구현
7   $('nav a').click(function (event) {
8       if (this.hash !== "") {
9           event.preventDefault();     // 기본 앵커 동작을 중단 (페이지가 즉시 이동하지 않게 함)
10          var hash = this.hash;       // 클릭한 링크의 해시값 저장
11          $('html, body').animate({   // html과 body 요소에 애니메이션 적용
12              scrollTop: $(hash).offset().top // 문서의 상단에서 해시값의 위치까지 스크롤
13          }, 800, function () {       // 애니메이션이 800밀리초(0.8초) 동안 진행됨
14              window.location.hash = hash;    // 스크롤이 끝나면 URL에 해시값 추가
15          });
16      }
17  });
18
19  // 로그인 모달 위치 업데이트 함수
20  function updateLoginModalPosition() {
21      var loginButtonOffset = $('#loginButton').offset();  // HTML에서 로그인 버튼의 위치와 크기 가져오기
22      var loginButtonWidth = $('#loginButton').outerWidth();
23      var modalWidth = $('#loginModal').outerWidth();      // HTML에서 로그인 모달의 너비 가져오기
24
25      $('#loginModal').css({ // 로그인 모달의 위치 설정
26          top: loginButtonOffset.top + $('#loginButton').outerHeight(),
27          left: loginButtonOffset.left + loginButtonWidth - modalWidth // 로그인 버튼의 오른쪽 끝에서 모달의 너비를 뺀 위치
28      });
29  }
30
```

전화번호 입력

```
100
101  // 전화번호 입력 시 자동 포커스 이동 및 숫자만 입력받기
102  $('#signup-phone-1, #signup-phone-2, #signup-phone-3').on('input', function () {
103      $(this).val($(this).val().replace(/\D/g, '')); // 현재 입력값에서 숫자만 남기기
104  });
105
106  // 전화번호 입력 시 자동 포커스 이동
107  $('#signup-phone-1').on('input', function () {
108      if ($(this).val().length >= 3) {    // 이 필드의 값의 길이가 3자 이상이면
109          $('#signup-phone-2').focus();   // 포커스를 #signup-phone-2 필드로 이동
110      }
111  });
112
113  $('#signup-phone-2').on('input', function () {
114      if ($(this).val().length >= 4) {
115          $('#signup-phone-3').focus();
116      } else if ($(this).val().length === 0) {  // 그렇지 않고, 이 필드의 값의 길이가 0자이면
117          $('#signup-phone-1').focus();         // 포커스를 다시 #signup-phone-1 필드로 이동
118      }
119  });
120
121  $('#signup-phone-3').on('input', function () {
122      if ($(this).val().length === 0) {
123          $('#signup-phone-2').focus();
124      }
125  });
126
```

슬라이더 설정

```
126
127     // 슬라이더 설정
128     // '[data-role="slider"]' 요소의 'data-width' 속성 값을 정수로 변환 ->  width 변수에 저장
129     var width = parseInt($('[data-role="slider"]').attr('data-width'));
130     var height = parseInt($('[data-role="slider"]').attr('data-height'));
131     var count = $('[data-role="slider"] .item').length; // '[data-role="slider"]' 내의 '.item' 요소들의 개수를 count 변수에 저장
132
133     // 첫 번째와 마지막 슬라이더 item 복제 -> 변수에 저장
134     var firstClone = $('[data-role="slider"] .item').first().clone();
135     var lastClone = $('[data-role="slider"] .item').last().clone();
136     // 복제한 item 요소를 '[data-role="slider"] .gallery'의 마지막, 처음에 추가
137     $('[data-role="slider"] .gallery').append(firstClone);
138     $('[data-role="slider"] .gallery').prepend(lastClone);
139
140     // 슬라이더 전체 크기 및 아이템 크기 설정
141     var totalItems = count + 2;       //원본 + 첫 번째 클론 + 마지막 클론
142     $('[data-role="slider"]').css({
143         width: width + 'px',          // 슬라이더의 너비 'width' 값으로 설정 (단위는 픽셀)
144         height: height + 'px'
145     }).find('.gallery').css({         // 내부 갤러리 CSS 적용
146         width: totalItems * width + 'px', // 갤러리의 너비를 'totalItems'의 수와 'width' 값을 곱한 값으로 설정
147         height: height + 'px',
148         display: 'flex',              // 갤러리의 디스플레이 스타일을 플렉스 박스로 설정
149         transform: 'translateX(' + -width + 'px)' // 갤러리의 초기 위치를 첫 번째 슬라이드로 설정
150     }).find('.item').css({            // 갤러리 안의 각 항목 CSS 적용
151         width: width + 'px',
152         height: height + 'px'
153     });
154
155     var currentPage = 1;      // 현재 페이지를 1로 초기화
156     var autoSlideInterval;    // 자동 슬라이드를 위한 변수 선언 (초기에는 슬라이드가 자동으로 이동하지 않음)
157
```

복제본 포함 순서:

[마지막 항목의 복제본 (3번 복제본)]

[1번 항목]

[2번 항목]

[3번 항목]

[첫 번째 항목의 복제본 (1번 복제본)]

슬라이더 설정

```
158     // 페이지 변경 함수
159     var changePage = function () {
160         $('[data-role="slider"] > .gallery').css({ // 해당 요소의 CSS 스타일을 변경
161             transition: 'transform 0.5s ease',      // 'transform' 속성의 전환 애니메이션을 설정 (0.5초 동안 부드럽게)
162             transform: 'translateX(' + (-currentPage * width) + 'px)' //현재 페이지에 맞게 이동 (왼쪽으로 currentPage * width 만큼 이동)
163         });
164     };
165 
166     // 슬라이더 애니메이션 종료 시 처리할 이벤트 핸들러 등록
167     $('[data-role="slider"] > .gallery').on('transitionend', function () {
168         if (currentPage === 0) {      // 만약 현재 페이지가 0이면 (즉, 마지막 항목의 복제본이면)
169             $('[data-role="slider"] > .gallery').css({
170                 transition: 'none',   // 전환 애니메이션을 제거
171                 transform: 'translateX(' + -(count * width) + 'px)' // 슬라이더를 실제 마지막 항목 위치로 이동
172             });
173             currentPage = count;      // 현재 페이지를 실제 마지막 항목으로 설정
174         }
175         // 만약 현재 페이지가 실제 항목의 개수 + 1이면 (즉, 첫 번째 항목의 복제본이면)
176         else if (currentPage === count + 1) {
177             $('[data-role="slider"] > .gallery').css({
178                 transition: 'none',
179                 transform: 'translateX(' + -width + 'px)' // 슬라이더를 실제 첫 번째 항목 위치로 이동
180             });
181             currentPage = 1;          // 현재 페이지를 실제 첫 번째 항목으로 설정
182         }
183     });
184 
```

슬라이더 설정

```
185     // 자동 슬라이드 함수
186     var autoSlide = function () {
187         // 다음 페이지로 이동하되, 페이지가 마지막 항목의 다음에 위치하면 실제 첫 번째 항목의 위치로 이동
188         currentPage = (currentPage + 1) % (count + 2);
189         changePage();  // 페이지 변경 함수 호출
190     };
191
192     // 자동 슬라이드 타이머 리셋 함수
193     var resetAutoSlideTimer = function () {
194         clearInterval(autoSlideInterval);   // 이전에 설정된 자동 슬라이드 타이머를 중지
195         // 새로운 자동 슬라이드 타이머를 설정 (3초마다 autoSlide 함수 실행)
196         autoSlideInterval = setInterval(autoSlide, 3000);
197     };
198
199     // 왼쪽 버튼 클릭 시 동작
200     $('#left-button').click(function () {
201             // 현재 페이지가 마지막 항목의 복제본이 아니면 이전 페이지로 이동
202         if (currentPage > 0) {
203             currentPage = currentPage - 1;
204         }   // 현재 페이지가 마지막 항목의 복제본이면 마지막 실제 항목으로 이동
205         else {
206             currentPage = count; // count : 실제 항목 개수 -> 실제 마지막 항목
207         }
208         changePage();           // 페이지 변경 함수 호출
209         resetAutoSlideTimer();  // 자동 슬라이드 타이머 재설정
210     });
211
```

슬라이더 설정

```
225     // 자동 슬라이드 시작
226     autoSlideInterval = setInterval(autoSlide, 3000);
227     // 현재 페이지의 슬라이더 항목에 'active' 클래스 추가
228     $('[data-role="slider"] .item').eq(currentPage).addClass('active');
229 });
230
231 // DOMContentLoaded 이벤트는 HTML 문서가 완전히 로드된 후 발생
232 document.addEventListener('DOMContentLoaded', function () {
233     // HTML안의 모든 내비게이션 버튼과 콘텐츠 요소를 선택
234     const buttons = document.querySelectorAll('.nav-button');
235     const contents = document.querySelectorAll('.content');
236
237     // 클래스가 'nav-button'이면서 data-target 속성이 'design'인 첫 번째 요소를 선택
238     const defaultButton = document.querySelector('.nav-button[data-target="design"]');
239     defaultButton.classList.add('active');      // 선택된 요소에 'active' 클래스를 추가
240     document.getElementById('design').classList.add('active'); // id가 'design'인 요소를 선택 후 'active' 클래스를 추가
241
242     // 네비게이션 버튼 클릭 시 처리
243     buttons.forEach(button => {     //버튼 목록을 반복하고 각 버튼에 대해 클릭 이벤트 리스너를 등록
244         button.addEventListener('click', () => {
245             //클릭된 버튼의 data-target 속성을 가져와서 변수 targetId에 저장
246             const targetId = button.getAttribute('data-target');
247
248             buttons.forEach(btn => {
249                 btn.classList.remove('active'); // 모든 버튼에서 'active' 클래스를 제거하여 이전에 활성화된 버튼의 표시를 해제
250             });
251             button.classList.add('active');     // 클릭된 버튼에만 'active' 클래스를 추가하여 클릭된 버튼을 활성 상태로 표시
252
253             contents.forEach(content => {   // 콘텐츠 목록을 반복하고, 각 콘텐츠의 id와 클릭된 버튼의 data-target 속성을 비교
254                 if (content.id === targetId) {
255                     content.classList.add('active'); 'active' // 일치 : 클래스를 추가하여 해당 콘텐츠를 활성 상태로 표시
256                 } else {
257                     content.classList.remove('active');       // 불일치 :  'active' 클래스를 제거하여 해당 콘텐츠를 비활성 상태로 변경
258                 }
```

개선점

저희는 대학 생활에서의 과제와 프로젝트를 효율적으로 관리하기 위한 도구를 개발했습니다.

이를 통해 강의 일정부터 과제, 그리고 팀 프로젝트까지 모두 체계적으로 관리할 수 있습니다.

학생들이 스트레스 없이 학업을 조직하고 효과적으로 협업할 수 있는 서비스를 제공하고 싶었습니다.