웹디자인 기말프로젝트

2423405 한서현

1. 문제 정의

중간프로젝트 컨텐츠의 디자인 버튼 등을 이용하여 변경 가능한 동적 페이지를 제작하면 된다.

2. html 코드에 관한 설명

① <style> 태그 부분

<style> 태그란 요소에 대한 인라인 CSS 스타일이다.

```
body {
    margin: 0;
    padding: 0;
    background-color: □white;
    color: ■black;
    font-family: 맑은 고딕, sans-serif;
}
```

margin: 0; -경계 주위의 영역이며, 웹페이지의 기본적인 여백을 제거한다. 이 프로그램에서는 이 값을 0으로 설정하여 페이지가 화면의 가장자리에 딱 맞게 표시되도록 할 때 사용된다.

padding: 0; - 컨텐츠 주위의 영역이며, 이 프로그램에서는 내부 여백을 없애서 body의 내용이 화면의 끝과 밀착되게 만들 때 사용된다.

background-color: white; - 배경색을 정의할 때 사용되며, 이 프로그램에서는 페이지의 배경색을 흰색으로 설정할 때 사용된다.

color: black; - 텍스트 색상을 변경할 때 사용되며, 이 프로그램에서는 텍스트의 기본 색상을 검은색으로 설정할 때 사용된다.

font-family: 맑은 고딕, sans-serif; - 폰트의 종류 속성이며, 이 프로그램에서는 글꼴을 맑은 고딕으로 설정하며, 만약 맑은 고딕이 없으면 sans-serif 글꼴을 사용하도록 할 때 사용된다.

```
header {
    background-color: ■black;
    color: □white;
    padding: 10px;
    text-align: center;
}
```

background-color와 color는 위에 설명되어 있기 때문에 생략

padding: 10px; - padding은 요소의 가장자리와 내용 간의 간격이며, 이 프로그램에서는 10px로 설정하여 헤더의 내용이 화면의 상단과 간격을 두고 표시되게 만들게 할 때 사용된다.

text-align: center; - 텍스트를 정렬할 때 사용되며, 이 프로그램에서는 텍스트를 가운데로 정렬되게 할 때 사용된다.

```
.container {
    width: 80%;
    margin: 20px auto;
}
```

width: 80%; - 너비를 설정할 때 사용되며, 이 프로그램에서는 .containter 클래스가 적용된 요소의 너비를 화면의 80%로 설정할 때 사용된다.

margin: 20px auto; - 경계 주위의 영역이며, 이 프로그램에서는 위와 아래 여백을 20픽셀로 설정하고, 좌우 여백을 auto로 설정하여 요소가 화면 가운데로 배치되게 할 때 사용된다.

```
h2 {
    color: ■black;
    border-bottom: 2px solid ■black;
}
```

color 관련 설명은 위에서 설명했기 때문에 생략

border-bottom: 2px solid black; - h2 태그 아래에 2px 두께의 검은색 실선 밑줄을 추가할 때 사용된다.

```
.famous img, .image-container img {
    width: 30%;
    border-radius: 8px;
    margin-right: 10px;
}
```

width 관련 설명은 위에서 설명했기 때문에 생략

border-radius: 8px; - 이미지의 모서리를 8px만큼 둥글게 처리할 때 사용된다.

margin-right: 10px; - 각 이미지의 오른쪽에 10px 간격을 두어 이미지 간 간격이 생기도록 할 때 사용된다.

```
.control-panel {
    margin: 20px auto;
    padding: 15px;
    border: 1px solid □#ddd;
    border-radius: 8px;
    width: 80%;
}
```

margin, padding, border-radius, width에 관련된 설명은 위에서 설명했기 때문에 생략

border: 1px solid #ddd; - 1px 두께의 연한 회색(#ddd) 실선 테두리를 추가하여 요소를 구분할 때 사용된다.

```
label {
    margin-right: 10px;
}
.input-group {
    margin-bottom: 15px;
}
```

margin-right에 관한 부분은 위에서 설명했기 때문에 생략

margin-bottom: 15px; - .input-group 하단에 15px의 여백을 추가해서 항목 간 간격을 넓혀줄 때 사용된다.

```
.memo-section {
    margin: 20px auto;
    padding: 15px;
    border: 1px solid #ddd;
    border-radius: 8px;
    width: 80%;
    background-color: #f9f9f9;
}
textarea {
    width: 100%;
    height: 50px;
}
```

이에 관한 설명은 위에서 전부 다 설명되었기 때문에 생략

height: 50px; - 텍스트 영역의 높이를 50px로 설정하여, 사용자가 여러줄을 입력할 수 있게 할 때 사용된다.

```
#memoDisplay {
    margin-left: 10px;
    font-weight: bold;
    color: green;
}
```

font-weight: bold; - 메모를 저장할 때 bold로 저장할 때 사용된다.

color: green; -메모를 저장할 때 초록색으로 저장할 때 사용된다.

② 제목 설정

```
<!-- 제목 설정 -->
<div class="control-panel">
    <h2>제목 설정</h2>
    <div class="input-group">
        <label for="titleColor">색상:</label>
        <input type="color" onchange="changeTitleColor(this.value)">
    </div>
    <div class="input-group">
        <label>스타일:</label>
        <input type="checkbox" onchange="changeTitleStyle('italic')"> italic
        <input type="checkbox" onchange="changeTitleStyle('bold')"> bold
    </div>
    <div class="input-group">
        <label>정렬:</label>
        <button onclick="changeTitleAlign('left')">왼쪽</button>
        <button onclick="changeTitleAlign('center')">가운데</button>
        <button onclick="changeTitleAlign('right')">오른쪽</button>
    </div>
    <div class="input-group">
        <label for="titleFont">모양(글꼴):</label>
        <select onchange="changeTitleFont(this.value)">
            <option value="굴림">굴림</option>
            <option value="맑은 고딕">맑은 고딕</option>
        </select>
    </div>
    <div class="input-group">
        <label for="titleFontSize">크기:</label>
        <input type="number" min="10" max="100" value="30" onchange="changeTitleSize(this.value)"> px
    </div>
    <button onclick="resetTitle()">초기화</button>
</div>
```

 태그: 논리적인 섹션을 생성할 때 사용된다.

<label> 태그: 입력 요소에 대한 레이블을 정의할 때 사용된다.

<input> 태그: 아주 다양하게 사용되는 입력 요소이다. type 속성에 따라 입력의 형태가 달라진다.

 : 다른 입력 요소들을 그룹화하기 위해 사용된다.

<button> 태그: 클릭 가능한 버튼을 생성한다.

<select> 태그: 드롭다운 리스트를 정의한다.

<option> 태그: 선택될 수 있는 항목을 정의한다.

 : class= "control-panel"은 이 에 "control-panel" 이라는 클래스를 부여하여 스타일을 적용할 수 있도록 한다.

 : 클래스에 "input-group"을 부여하여 이 안에 포함된 입력 요소들이 그룹화된다. 이 그룹화된 입력 요소들을 스타일링하거나 조작할 때 유용하다.

<label for="titleColor">색상:</label> : for="titlecolor"는 이 레이블이 "titlecolor"라는 id를 가진 요

소와 연결됨을 나타내며, 색상: 은 사용자가 색상을 선택할 수 있다는 것을 인식하게 해준다.

<input type="color" onchange="changeTitleColor(this.value)"> : input type="color"는 사용자가 색상을 선택할 수 있게 해주며, onchange="changeTitleColor(this.value)"는 사용자가 색상을 선택할 때 마다 changeTitleColor라는 함수가 호출되도록 한다. 이 함수는 선택한 색상을 this.value로 받아서 제목 색상을 변경하는 역할을 한다.

<label>스타일:</label> : 사용자가 스타일을 선택할 수 있다는 것을 인식할 수 있게 해준다.

<input type="checkbox" onchange="changeTitleStyle('italic')"> italic : input type="checkbox"는 체크박스를 표시하며, 사용자가 이 체크박스를 선택하면 onchange="changeTitleStyle('italic') 이 함수가 사용이 된다. 이 함수는 제목의 스타일을 italic(기울임)로 변경할 수 있게 한다. 텍스트로 italic로 작성된 부분은 사용자가 체크박스를 선택했을 때 italic로 적용된다는 것을 인식할 수 있게 해준다.

<input type="checkbox" onchange="changeTitleStyle('bold')"> bold : <input type="checkbox" onchange="changeTitleStyle('italic')"> italic 이 부분과 설명이 동일하며, 사용자가 bold를 선택하면 제목을 bold(굵게) 변경할 수 있게 해준다.

<label>정렬:</label> : 사용자가 제목을 정렬할 수 있는 버튼이 있다는 것을 알려준다.

<button onclick="changeTitleAlign('left')">왼쪽</button> : onclick="changeTitleAlign('left')는 사용자가 이 버튼을 클릭할 때 마다 changeTitleAlign('left') 함수가 호출되서 제목을 왼쪽 정렬로 변경한다. 텍스트 왼쪽은 왼쪽이라는 버튼이 있다는 것을 사용자가 인식할 수 있게 해준다.

<button onclick="changeTitleAlign('center')">가운데</button> : changeTitleAlign('center')는 사용자가 이 버튼을 클릭할 때 마다 changeTitleAlign('center') 함수가 호출되서 제목을 가운데 정렬로 변경한다. 텍스트 가운데는 가운데라는 버튼이 있다는 것을 사용자가 인식할 수 있게 해준다.

<button onclick="changeTitleAlign('right')">오른쪽</button> : changeTitleAlign('right')는 사용자가 이 버튼을 클릭할 때 마다 changeTitleAlign('right') 함수가 호출되서 제목을 오른쪽 정렬로 변경한다. 텍스트 오른쪽은 오른쪽이라는 버튼이 있다는 것을 사용자가 인식할 수 있게 해준다.

<label for="titleFont">모양(글꼴):</label> : 모양(글꼴): 이라는 텍스트를 통해 모양(글꼴) 이라는 것이 있다는 것을 사용자가 인식할 수 있게 해준다.

<select onchange="changeTitleFont(this.value)"> : onchange="changeTitleFont(this.value)" 는 사용자가 드롭다운에서 폰트를 선택할 때 마다 changeTitleFont 함수가 호출되어 제목의 글꼴을 변경할 수 있게 해준다.

<option value="굴림">굴림</option> : 텍스트 굴림은 사용자가 굴림이라는 글꼴을 선택할 수 있게 해준다. 이 값은 changeTitleFont 함수에서 사용된다.

<option value="맑은 고딕">맑은 고딕</option> : 텍스트 굴림은 사용자가 맑은고딕 이라는 글꼴을 선택할 수 있게 해준다.

<label for="titleFontSize">크기:</label>: 텍스트 크기는 제목의 글꼴 크기를 설정할 수 있는 부분을 설명하는 텍스트이다.

<input type="number" min="10" max="100" value="30" onchange="changeTitleSize(this.value)"> px : <input type="number"> 는 숫자 입력을 받는 필드를 생성하고, min="10" max="100"은 최소값 10, 최대값 100으로 제목 크기를 제한하고, value="30 은 기본값을 30으로 설정하게 해주고, onchange="changeTitleSize(this.value)"는 사용자가 크기를 변경할 때 마다 changeTitleSize 함수가 호출되어 제목의 크기가 조정할 수 있게 해준다.

<button onclick="resetTitle()">초기화</button> : onclick="resetTitle()"은 사용자가 버튼을 클릭할 때 마다 resetTitle()이라는 함수가 호출되도록 한다. 이 함수는 제목의 스타일을 초기 상태로 되돌리게 하는 역할을 하고 있다. 예로 들면 색상, 폰트, 크기 등을 원래 값으로 되돌리게 한다.

③ 내용 설정

```
<!-- 내용 설정 -->
<div class="control-panel">
    <h2>내용 설정</h2>
    <div class="input-group">
        <label for="contentColor">색상:</label>
        <input type="color" onchange="changeContentColor(this.value)">
    </div>
    <div class="input-group">
        <label>스타일:</label>
        <input type="checkbox" onchange="changeContentStyle('italic')"> italic
        <input type="checkbox" onchange="changeContentStyle('bold')"> bold
    </div>
    <div class="input-group">
        <label>정렬:</label>
        <button onclick="changeContentAlign('left')">왼쪽</button>
        <button onclick="changeContentAlign('center')">가운데</button>
        <button onclick="changeContentAlign('right')">오른쪽</button>
    </div>
    <div class="input-group">
        <label for="contentFont">모양(글꼴):</label>
        <select onchange="changeContentFont(this.value)">
            <option value="굴림">굴림</option>
            <option value="맑은 고딕">맑은 고딕</option>
        </select>
    </div>
    <div class="input-group">
        <label for="contentFontSize">크기:</label>
        <input type="number" min="10" max="100" value="30" onchange="changeContentSize(this.value)"> px
    </div>
    <button onclick="resetContent()">초기화</button>
</div>
```

내용설정은 제목설정과 내용이 거의 동일하기 때문에 설명은 생략한다. 둘의 차이점은 함수가 Tilte인 것과 Content인 것의 차이말고는 내용이 동일하다.

④ 콘텐츠

```
<!-- 콘텐츠 -->
<div class="container">
    <section class="recommendation">
        <h2>1. 롱코트와 스웨터 조합</h2>
        <img src="longcoatandsweater.jpg.png" alt="롱코트와 스웨터">
        <p>롱코트는 전체적인 스타일과 우아함을 더해줍니다.</p>
        <p>사진 및 내용 출처: <a href="https://c-writing.aztext.net/entry/%EA%B0%80%EC%9D%84-%ED%8C%A8%EC%85%98-%EC%BD%94%EB%94%94-%EB%B8%94%EB%9E%99-%ED%95%98%EC%9D%B4%EC%98%A8%EC%9D%B4%EC%8A%A4%ED%8A%B8-%ED%
            가을 패션 코디 블랙 하이웨이스트 팬츠에 터틀넥 스웨터와 롱코트 中 모던시크룩</a></p>
    </section>
```

```
        <section class="recommendation">
            <h2>2. 무스탕과 기모 조거팬츠</h2>
            <img src="mustangandjokerpants.jpg" alt="무스탕과 기모 조거팬츠">
            <p>무스탕과 기모 조거팬츠를 조합하여 트렌디 하면서 따뜻하게 입을 수 있다. 또한 개인 취향에 따라 어그부츠를 추가할 수 있다.</p>
            <p>사진 및 내용 출처: <a href="https://v.daum.net/v/GAkt48D7r4" target="_blank">겨울철 따뜻한 옷들로 해본 패셔너블한 코디3 中 크롭 무스탕 & 기모 조거팬츠</a></p>
        </section>
        <section class="recommendation">
            <h2>3. 니트와 롱스커트</h2>
            <img src="neatandlongskirt.jpg" alt="니트와 롱스커트">
            <p>롱스커트와 니트를 함께 입으면 보온성과 함께 세련미를 동시에 추구할 수 있다.</p>
            <p>사진 출처: <a href="https://v.daum.net/v/5bfe5188f3a1d400013884ec" target="_blank">겨울철 따뜻한 여자 롱스커트 코디 이렇게 입어봐★</a></p>
        </section>
        <section class="recommendation">
            <h2>4. 겨울용 가디건</h2>
            <img src="gardigeon.jpg" alt="겨울용 가디건">
            <p>겨울용 가디건은 어느 옷에나 입어도 잘 어울리기 때문에 스타일에 따라 다양한 분위기들을 연출할 수 있다.</p>
            <p>사진 출처: <a href="http://www.10x10.co.kr/shopping/category_prd.asp?itemid=3483884" target="_blank">여성 브이넥 꽈배기 니트 울소재 겨울 가디건</a></p>
        </section>
        <section class="recommendation">
            <h2>5. 액세서리 활용</h2>
            <div class="image-container">
                <img src="bini.jpg" alt="겨울용 모자들">
                <img src="muffler.jpg" alt="머플러">
                <img src="gloves.jpg" alt="겨울용 장갑">
            </div>
            <p>모자, 머플러, 장갑 등의 액세서리는 보온성과 스타일을 동시에 제공한다.</p>
            <p>사진 출처: <a href="https://post.naver.com/viewer/postView.naver?volumeNo=30531703&memberNo=4656129" target="_blank">
                겨울 코디엔 포근함 한 스푼 추가요! 겨울 악세사리 모음.zip</a></p>
        </section>
    </div>
```

 태그는 위에서 설명했기 때문에 생략

<section> 태그: 문서의 섹션을 의미한다.

<img> 태그: 이미지를 삽입하는 태그

<section class="recommendation"> : 각 코디 항목을 구분하는 <section> 태그이다. class="recommendation"을 통해 해당 섹션을 스타일링 할 수 있다. 각 섹션은 다른 코디에 대한 정보를 담고 있다.

<h2>1. 롱코트와 스웨터 조합</h2> : <h2> 태그는 제목을 나타내는 태그이다.

<img src="longcoatandsweater.jpg.png" alt="롱코트와 스웨터"> : src="longcoatandsweater.jpg.png"은 이미지 파일의 경로이며, alt="롱코트와 스웨터"는 이미지가 나타내는 내용을 설명하는 텍스트이다.

롱코트는 전체적인 스타일과 우아함을 더해줍니다. : 태그는 문단을 나타내는 태그

이다.

사진 및 내용 출처: 가을 패션 코디 블랙 하이웨이스트 팬츠에 터틀넥 스웨터와 롱코트 中 모던시크룩 : 코디에 대한 설명과 출처를 제공할 때 사용된다.

나머지 내용들은 위의 설명과 비슷하기 때문에 생략

```
<div class="memo-section">
    <h2>메모 추가</h2>
    <textarea id="memoText" placeholder="메모를 추가하세요..."></textarea>
    <button onclick="saveMemo()">메모 저장</button>
    <span id="memoDisplay"></span> <!-- 저장된 메모가 표시될 부분 -->
</div>
```

 태그는 "메모 추가" 섹션을 감싸는 요소이며, class="memo-section"은 이 div에 특정 스타일을 적용하기 위해 클래스 이름을 설정한 것이며, <textarea id="memoText" placeholder="메모를 추가하세요..."></textarea>는 <textarea> 태그는 여러 줄의 텍스트를 입력할 수 있는 영역을 만들어주고, 사용자가 메모를 입력할 수 있도록 해주며, id="memoText"는 이 텍스트의 고유 id를 설정하는 것이고, placeholder="메모를 추가하세요..." 는 텍스트에 아무것도 입력되지 않았을 때 보여주는 메시지이다.

⑤ 제목 스타일링 함수

```
<script>
    let titleElement = document.querySelector('header h1');
    let titleElements = document.querySelectorAll('h2');
    let contentElements = document.querySelectorAll('.recommendation p');

    // 제목 스타일링 함수
    function changeTitleColor(color) {
        titleElement.style.color = color;
        titleElements.forEach(function(title) {
            title.style.color = color;
        });
    }

    function changeTitleStyle(style) {
        if (style === 'italic') {
            titleElement.style.fontStyle = titleElement.style.fontStyle === 'italic' ? 'normal' : 'italic';
            titleElements.forEach(function(title) {
                title.style.fontStyle = title.style.fontStyle === 'italic' ? 'normal' : 'italic';
            });
        } else if (style === 'bold') {
            titleElement.style.fontWeight = titleElement.style.fontWeight === 'bold' ? 'normal' : 'bold';
            titleElements.forEach(function(title) {
                title.style.fontWeight = title.style.fontWeight === 'bold' ? 'normal' : 'bold';
            });
        }
    }

    function changeTitleAlign(align) {
        titleElement.style.textAlign = align;
        titleElements.forEach(function(title) {
            title.style.textAlign = align;
        });
    }

    function changeTitleFont(font) {
        titleElement.style.fontFamily = font;
        titleElements.forEach(function(title) {
            title.style.fontFamily = font;
        });
    }

function changeTitleSize(size) {
    titleElement.style.fontSize = size + 'px';
    titleElements.forEach(function(title) {
        title.style.fontSize = size + 'px';
    });
}

function resetTitle() {
    titleElement.style = '';
    titleElements.forEach(function(title) {
        title.style = '';
    });
}
```

let titleElement = document.querySelector('header h1'); : header 태그 내의 h1 요소를 선택하여 titleElement 변수에 저장한다. 이는 페이지의 제목을 나타내는 h1 요소이다.

let titleElements = document.querySelectorAll('h2'); : 모든 h2 요소들을 선택하여 titleElements 배열에 저장한다.

let contentElements = document.querySelectorAll('.recommendation p'); : .recommendation 클래스를 가진 모든 요소 내의 p태그들을 선택하여 contentElements 배열에 저장한다.

```
function changeTitleColor(color) {
    titleElement.style.color = color;
    titleElements.forEach(function(title) {
        title.style.color = color;
    });
}
```

이 함수는 <h1>과 <h2>의 글자 색상을 변경하는 함수이다.

titleElement.style.color = color;를 통해 header 안에 있는 <h1> 태그의 색상을 color 매개변수로 전달된 값으로 변경하며, titleElements.forEach(function(title)를 통해 titleElements 배열에 포함된 모든 h2 태그에 대해 반복문을 실행하며, title.style.color = color;를 통해 각 <h2> 태그의 색상도 동일하게 color로 변경한다.

```
function changeTitleStyle(style) {
    if (style === 'italic') {
        titleElement.style.fontStyle = titleElement.style.fontStyle === 'italic' ? 'normal' : 'italic';
        titleElements.forEach(function(title) {
            title.style.fontStyle = title.style.fontStyle === 'italic' ? 'normal' : 'italic';
        });
    } else if (style === 'bold') {
        titleElement.style.fontWeight = titleElement.style.fontWeight === 'bold' ? 'normal' : 'bold';
        titleElements.forEach(function(title) {
            title.style.fontWeight = title.style.fontWeight === 'bold' ? 'normal' : 'bold';
        });
    }
}
```

이 함수는 제목의 글꼴 스타일(italic, bold)을 변경할 때 사용되는 함수이다.

if (style === 'italic') : 만약 스타일 매개변수가 'italic' 이라면 titleElement.style.fontStyle = titleElement.style.fontStyle === 'italic' ? 'normal' : 'italic'; : <h1> 태그의 글꼴 스타일이 italic 이면 normal로, 그렇지 않으면 italic로 변경하며, titleElements.forEach(function(title) 은 <h2> 태그들도 동일하게 italic 스타일로 한다.

bold 부분은 italic에서 설명했던 부분들을 bold로 변경하면 된다. 그래서 설명은 생략한다.

```
function changeTitleAlign(align) {
    titleElement.style.textAlign = align;
    titleElements.forEach(function(title) {
        title.style.textAlign = align;
    });
}
```

이 함수는 제목을 정렬할 때 사용되는 함수이다.

titleElement.style.textAlign = align; 를 통해 <h1> 태그의 텍스트 정렬을 align 매개변수 값으로 설정하며, titleElements.forEach(function(title)를 통해 <h2> 태그들의 텍스트 정렬도 동일하게 설

정하며, title.style.textAlign = align; 를 통해 각 <h2> 태그의 텍스트 정렬을 align 값으로 설정할 때 사용한다.

```
function changeTitleFont(font) {
    titleElement.style.fontFamily = font;
    titleElements.forEach(function(title) {
        title.style.fontFamily = font;
    });
}
```

이 함수는 제목의 글꼴을 변경할 때 사용되는 함수이다.

titleElement.style.fontFamily = font; 를 통해 <h1> 태그의 글꼴을 font 매개변수의 값으로 설정하며, (ex: 맑은고딕, 굴림), titleElements.forEach(function(title)를 통해 <h2> 태그들의 글꼴도 동일하게 설정하며, title.style.fontFamily = font; 를 통해 각 <h2>의 태그의 글꼴을 font 값으로 설정한다.

```
function changeTitleSize(size) {
    titleElement.style.fontSize = size + 'px';
    titleElements.forEach(function(title) {
        title.style.fontSize = size + 'px';
    });
}
```

이 함수는 제목의 글꼴 크기를 변경할 때 사용하는 함수이다.

titleElement.style.fontSize = size + 'px'; 는 <h1> 태그의 글꼴 크기를 size 매개변수 값에 'px'를 추가하여 설정하며, titleElements.forEach(function(title) 는 <h2> 태그들의 글꼴 크기도 동일하게 설정하며, title.style.fontSize = size + 'px'; 는 각 <h2> 태그의 글꼴 크기를 size+'px' 로 설정할 때 사용된다.

```
function resetTitle() {
    titleElement.style = '';
    titleElements.forEach(function(title) {
        title.style = '';
    });
}
```

이 함수는 제목에 적용된 모든 스타일을 초기화할 때 사용되는 함수이다.

titleElement.style = ''; 은 <h1> 태그의 모든 인라인 스타일을 제거하며, titleElements.forEach(function(title)은 <h2> 태그에 대해서 동일한 스타일 초기화를 수행할 때 사용되며, title.style = ''; 은 각 <h2> 태그의 모든 인라인 스타일을 제거할 때 사용된다. 이렇게 하면 제목에 대한 사용자 설정이 모두 초기화가 된다.

⑥ 내용 스타일링 함수

```
// 내용 스타일링 함수
function changeContentColor(color) {
    contentElements.forEach(function(content) {
        content.style.color = color;
    });
}

function changeContentStyle(style) {
    contentElements.forEach(function(content) {
        if (style === 'italic') {
            content.style.fontStyle = content.style.fontStyle === 'italic' ? 'normal' : 'italic';
        } else if (style === 'bold') {
            content.style.fontWeight = content.style.fontWeight === 'bold' ? 'normal' : 'bold';
        }
    });
}

function changeContentAlign(align) {
    contentElements.forEach(function(content) {
        content.style.textAlign = align;
    });
}

function changeContentFont(font) {
    contentElements.forEach(function(content) {
        content.style.fontFamily = font;
    });
}

function changeContentSize(size) {
    contentElements.forEach(function(content) {
        content.style.fontSize = size + 'px';
    });
}

function resetContent() {
    contentElements.forEach(function(content) {
        content.style = '';
    });
}
```

```
function changeContentColor(color) {
    contentElements.forEach(function(content) {
        content.style.color = color;
    });
}
```

이 함수는 .recommendation 클래스의 p 태그들(본문 내용)의 글자 색상을 변경할 때 사용하는 함수이다.

contentElements.forEach(function(content)를 통해 contentElements 배열에 포함된 모든 요소 (p 태그)에 대해 반복문을 실행하며, content.style.color = color; 를 통해 각 요소(content)의 스타일을 변경하여 텍스트 색상을 color 매개변수로 전달된 값으로 설정한다.

```
function changeContentStyle(style) {
    contentElements.forEach(function(content) {
        if (style === 'italic') {
            content.style.fontStyle = content.style.fontStyle === 'italic' ? 'normal' : 'italic';
        } else if (style === 'bold') {
            content.style.fontWeight = content.style.fontWeight === 'bold' ? 'normal' : 'bold';
        }
    });
}
```

이 함수는 본문 내용의 글꼴 스타일을 변경할 때 사용되며, italic 또는 bold 스타일로 변경이 가능하다.

contentElements.forEach(function(content)를 통해 contentElements 배열에 포함된 모든 p 태그에 대해 반복문을 실행하며, if (style === 'italic')은 스타일 매개변수가 'italic' 일 때, content.style.fontStyle = content.style.fontStyle === 'italic' ? 'normal' : 'italic'; 은 만약 현재 글꼴 스타일이 italic이면 normal로 바꾸고, 그렇지 않으면 italic로 설정하며, else if (style === 'bold'): 스타일 매개변수가 'bold' 일때, content.style.fontWeight = content.style.fontWeight === 'bold' ? 'normal' : 'bold'; : 만약 현재 글꼴 두께가 bold 이면 normal로 바꾸고, 그렇지 않으면 bold로 설정한다.

```
function changeContentAlign(align) {
    contentElements.forEach(function(content) {
        content.style.textAlign = align;
    });
}
```

이 함수는 본문 내용의 텍스트의 정렬을 변경할 때 사용된다.

contentElements.forEach(function(content) 를 통해 contentElements 배열에 포함된 모든 p 태그에 대해 반복문을 실행하며, content.style.textAlign = align;를 통해 각 p 태그의 textAlign 스타일을 align 매개변수로 전달된 값('left', 'center', 'right')로 설정한다.

```
function changeContentFont(font) {
    contentElements.forEach(function(content) {
        content.style.fontFamily = font;
    });
}
```

이 함수는 본문 내용의 글꼴을 변경할 때 사용된다.

contentElements.forEach(function(content)를 통해 contentElements 배열에 포함된 모든 p 태그에 대해 반복문을 실행하며, content.style.fontFamily = font;를 통해 각 p 태그의 fontFamily 스타일을 매개변수로 전달된 값(ex: 굴림, 맑은고딕)으로 설정한다.

```
function changeContentSize(size) {
    contentElements.forEach(function(content) {
        content.style.fontSize = size + 'px';
    });
}
```

이 함수는 본문 내용의 글꼴 크기를 변경할 때 사용된다.

contentElements.forEach(function(content)를 통해 contentElements 배열에 포함된 모든 p 태그에 대해 반복문을 실행하며, content.style.fontSize = size + 'px'; 를 통해 각 p 태그의 fontSize 스타일을 size 매개변수로 전달된 값에 'px' 단위를 붙여 설정한다. ex: size=16일 경우 fontSize='16px'로 설정된다.

```
function resetContent() {
    contentElements.forEach(function(content) {
        content.style = '';
    });
}
```

이 함수는 본문 내용의 모든 스타일을 초기화하는 함수이다.

contentElements.forEach(function(content)를 통해 contentElements 배열에 포함된 모든 p 태그에 대해 반복문을 실행하며, content.style = '';를 통해 각 p 태그의 style 속성을 빈 문자열로 설정하여 모든 인라인 스타일을 제거한다.

⑦ 메모 저장 함수

```
// 메모 저장 함수
function saveMemo() {
    let memoText = document.getElementById('memoText').value;
    localStorage.setItem('memo', memoText);
    document.getElementById('memoDisplay').innerText = memoText;
}

window.onload = function() {
    let savedMemo = localStorage.getItem('memo');
    if (savedMemo) {
        document.getElementById('memoText').value = savedMemo;
        document.getElementById('memoDisplay').innerText = savedMemo;
    }
}
```

function saveMemo() { : 사용자가 메모를 저장하는 기능을 하는 함수이며, 이 함수는 메모를 텍스트 상자에 입력 후 메모 저장 버튼을 클릭하였을 때 호출이 된다.

let memoText = document.getElementById('memoText').value; :

document.getElementById('memoText')는 id가 memText 인 요소를 찾으며, .value는 해당 텍스트 영역에서 사용자가 입력한 텍스트 값을 가져오고, 이 값을 memoText 변수에 저장한다.

localStorage.setItem('memo', memoText); : 웹브라우저의 로컬스토리지에 데이터를 저장하는 메소드이며, 'memo'는 저장될 키(key)로, 메모를 식별할 수 있게 해주고, memoText는 위에서 가져온 메모 텍스트 값으로, 이 값이 'memo'라는 키에 저장된다.

document.getElementById('memoDisplay').innerText = memoText; : id가 memoDisplay요소를 찾으며, .innerText는 해당 요소의 텍스트 내용을 수정할 때 사용되고, memoText 변수에 저장된 메모 텍스트를 memoDisplay 요소에 표시한다. 메모를 저장한 후 사용자가 입력한 메모가 페이지에 바로 표시가 된다.

window.onload = function() { : window.onload는 웹페이지가 모드 로드되었을 때 실행되는 이벤트이며, 페이지가 처음 열리거나 새로고침 될 때 실행이 된다.

let savedMemo = localStorage.getItem('memo'); 로컬 스토리지에서 'memo'라는 키로 저장된 값을 가져오며, 가져온 값은 savedMemo변수에 저장이 된다.

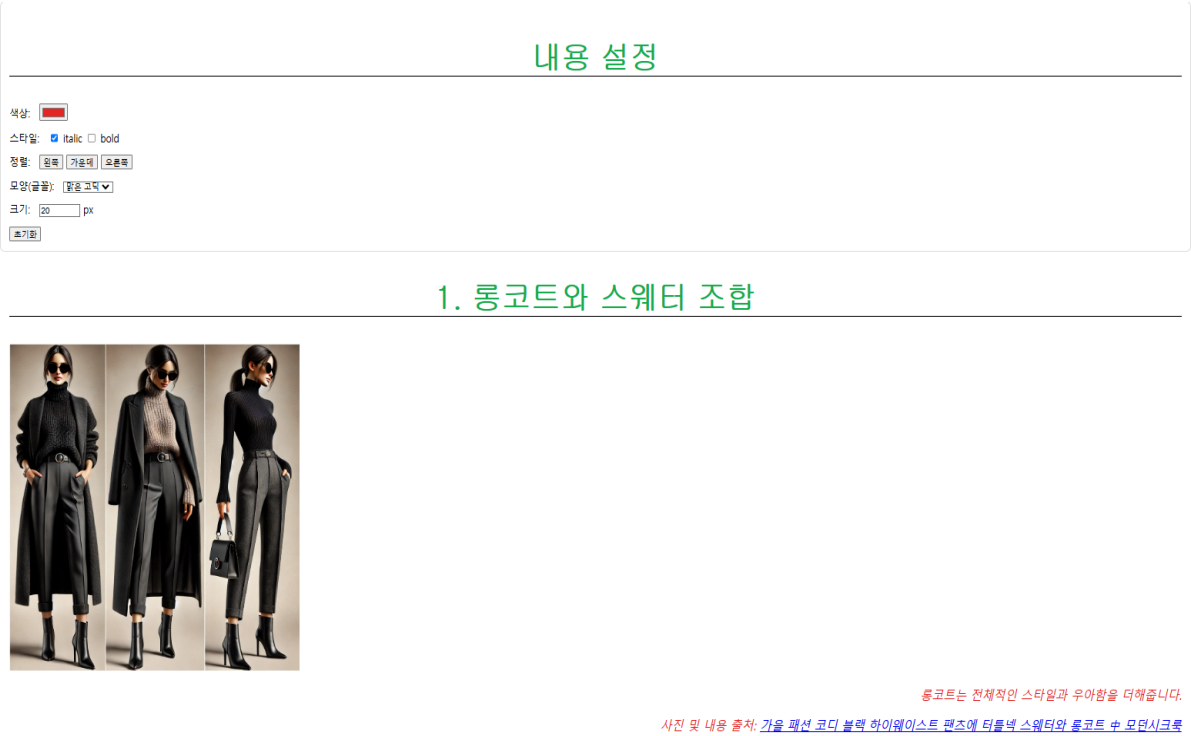
```
if (savedMemo) {
    document.getElementById('memoText').value = savedMemo;
    document.getElementById('memoDisplay').innerText = savedMemo;
}
```

savedMemo 값이 존재하는지, 즉 로컬 스토리지에 메모가 저장되었는지 확인하는 조건문이며, 만약 로컬 스토리지에 저장된 메모가 있다면 document.getElementById('memoText').value = savedMemo; 는 텍스트 영역에 저장된 메모를 표시하고, document.getElementById('memoDisplay').innerText = savedMemo;는 페이지 상의 span()요소에 저장된 메모를 표시한다.

3. 최종 결과

제목설정 하였을 때(초록색, bold, 가운데정렬, 굴림, 폰트크기 50px 기준)

내용설정 하였을 때(빨간색, italic, 오른쪽 정렬, 맑은 고딕, 20px)



메모 추가 (안녕하세요 라는 메모 저장)

더 자세한 결과는 따로 pdf파일을 첨부함

4. 전체 코드

전체 코드는 따로 pdf파일을 첨부함