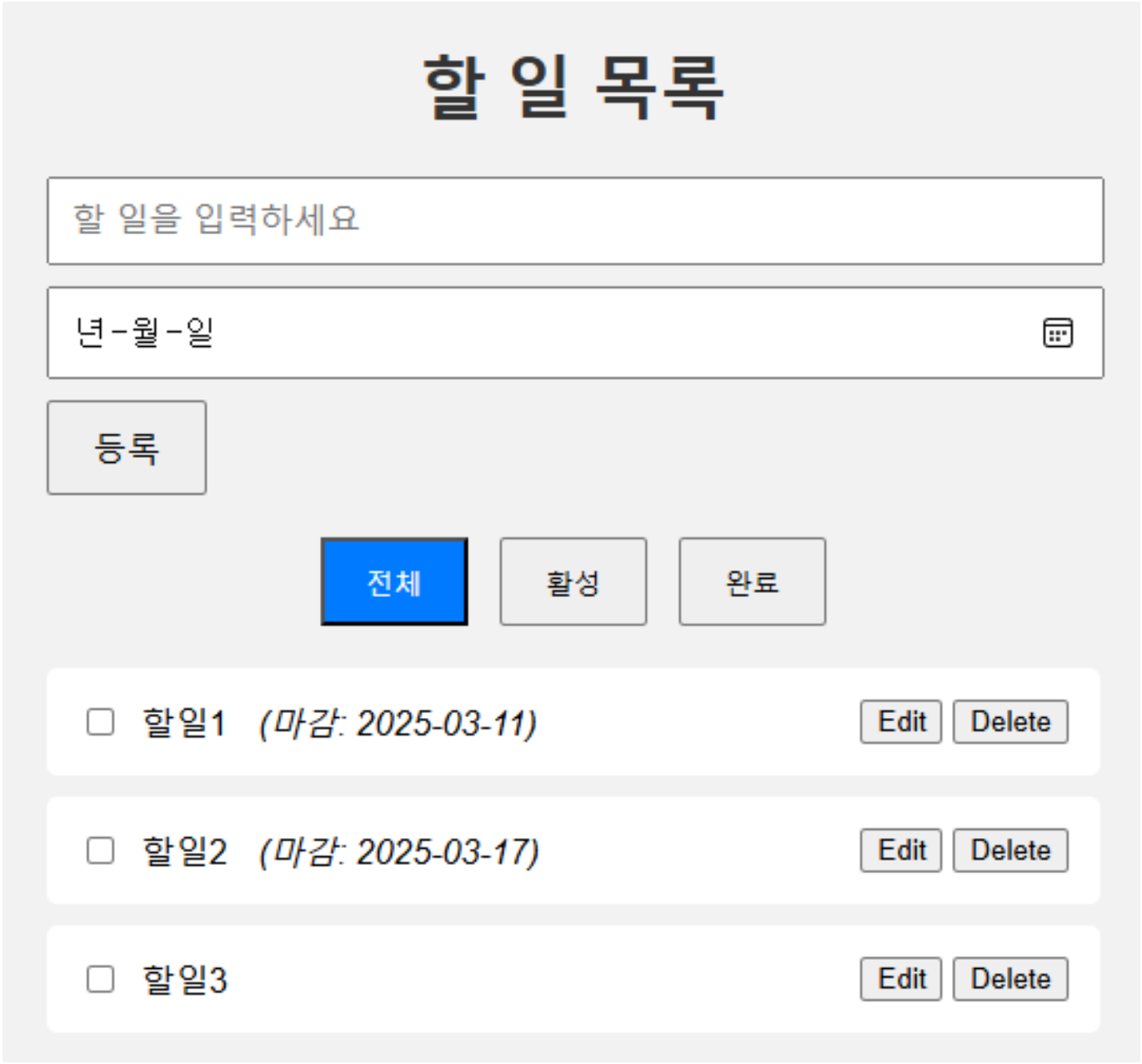

인터랙티브 할 일 관리(App) 만들기

과제 개요

순수 JavaScript(즉, 외부 라이브러리 없이)로 "할 일 관리" 웹 애플리케이션을 제작합니다. 이 앱은 사용자가 할 일을 추가, 편집, 삭제하고 완료 여부를 토글하는 기능을 제공합니다. 또한, 사용자가 현재 할 일 목록을 "전체/활성/완료" 필터로 분류하여 볼 수 있도록 합니다. (추가적으로 로컬 스토리지를 활용하면 앱 종료 후에도 데이터가 유지되도록 할 수 있습니다.) 이 과제는 DOM 객체를 이용한 엘리먼트 생성, 이벤트 핸들링, 동적 클래스 조작 등 다양한 JavaScript 기법들을 연습할 수 있습니다.

세부 요구사항

1. #### 기본 인터페이스 구성

- 헤더 및 제목:

  - 페이지 상단에 "할 일 목록" 혹은 "To-Do List"라는 제목을 나타냅니다.

- 할 일 추가 폼:

  - 입력 필드: 할 일의 제목을 입력할 수 있는 텍스트 필드. (빈 값일 때는 추가되지 않도록 예외 처리)

  - 날짜 입력: (선택 사항) 할 일의 마감 기한을 입력할 수 있도록 합니다.

  - 등록 버튼: "Add Task" 혹은 "등록" 버튼을 눌러 새로운 할 일을 목록에 추가합니다.

  - 에러 메시지: 입력 필드가 빈 값인 경우, 화면에 경고 메시지를 표시하고 몇 초 후 사라지게 합니다.

2. #### 동적 할 일 목록

- 폼 제출 후, 새로운 할 일이 리스트(li 또는 div) 항목으로 추가됩니다. 각 항목에는 다음과 같은 요소들이 포함되어야 합니다.

  - 할 일 설명: 사용자가 입력한 텍스트를 표시합니다.

  - 마감 날짜: 입력된 날짜(있는 경우)를 함께 표시합니다.

  - 상태 표시 및 토글: 체크박스나 버튼을 이용하여 할 일의 상태(완료/미완료)를 변경할 수 있어야 합니다.

    - 완료된 할 일은 시각적으로 구분(예: 텍스트에 취소선, 색상 변화 등)되어야 합니다.

  - 액션 버튼: 각 할 일 항목에 대해

    - Edit 버튼: 클릭 시 해당 할 일을 인라인(in-line)으로 편집할 수 있는 입력 필드로 변경하고, 수정 완료 후 저장.

    - Delete 버튼: 클릭 시 해당 할 일을 목록에서 삭제.

3. #### 필터링 기능

- 화면 상의 별도 영역(예: 버튼 그룹 혹은 탭)에서 "전체", "활성", "완료" 상태에 따라 할 일 목록을 필터링 할 수 있어야 합니다.

  - 전체(All): 모든 할 일을 표시.

  - 활성(Active): 완료되지 않은 할 일만 표시.

- 완료(Completed): 완료된 할 일만 표시.

- 필터 버튼 클릭 시, 현재 목록 DOM을 동적으로 업데이트하여 해당하는 할 일들만 보여주어야 합니다.

4. #### DOM 조작 및 이벤트 처리

- 모든 할 일 항목은 `document.createElement` 메서드를 활용하여 동적으로 생성합니다.

- 이벤트 리스너를 각 버튼(등록, Edit, Delete, 상태 토글) 또는 부모 요소에 이벤트 위임(Delegation) 방식을 활용하여 등록합니다.

- DOM 클래스 조작 (`classList.add`, `classList.remove`, `classList.toggle`)을 이용해 스타일 변화(예: 완료된 할 일에 취소선 적용)를 구현합니다.

5. #### 레이아웃 및 스타일

- CSS 활용: 기본적인 CSS를 활용하여 레이아웃을 구성합니다. (예: Flexbox나 Grid를 사용해 반응형 디자인 고려)

- 사용자 경험(UX):

  - 할 일 추가 후 입력 필드를 자동으로 초기화합니다.

  - 클릭한 버튼의 피드백(hover 효과, active 상태)을 구현합니다.

- 접근성: 요소에 적절한 `aria-*` 속성이나 시맨틱 태그들을 활용하여 접근성을 고려합니다.

6. #### 추가(보너스) 요구사항

- 로컬 스토리지(Local Storage) 활용: 페이지 새로고침 시에도 할 일 데이터가 유지되도록 저장, 불러오기 기능을 구현합니다.

- 유효성 검사 강화: 할 일 제목의 길이 제한, 중복 방지 등 추가적인 예외 처리를 구현합니다.

- 애니메이션 효과: 할 일 항목의 추가/삭제 시 간단한 효과(예: fade in/out)를 적용하여 사용자 경험을 향상시킬 수 있습니다.

- 코드 구조화: 함수, 모듈 또는 클래스를 적절히 사용하여 코드의 가독성을 높입니다.

프로젝트 진행 시 고려할 사항

- 문제 분해:

  전체 프로젝트를 여러 컴포넌트(입력 폼, 할 일 목록, 필터 컨트롤 등)로 분리한 후, 각 컴포넌트의 역할과 이벤트 흐름을 도식화해 보세요.

- 이벤트 플로우:

  ```
  사용자
    |
    ▼
  입력 폼(할 일 입력 및 등록 버튼)
    | (등록 버튼 클릭)
    ▼
  할 일 목록에 새로운 항목 생성 (DOM 생성)
    | ──────────────────► 각 항목의 Edit/ Delete/ Toggle 이벤트 처리
    ▼
  필터 버튼 클릭 → 목록 필터링
  ```

- 디버깅 및 테스트:

  각 기능별로 작은 단위 테스트를 진행하면서, 브라우저 개발자 도구를 사용해 콘솔 에러나 논리적 문제를 해결해 나갑니다.

- 설계 다이어그램:

  아래의 간단한 ASCII 다이어그램을 참고하여 전체 애플리케이션의 데이터 흐름을 구상해 보세요.

  ```
  [Header]
    |
  ```

```
        ▼
    [할 일 입력 폼] ——> (입력/등록) ——> [할 일 항목 생성]
        |                                   |
        |                                   ├——> [상태 토글]
        |                                   ├——> [Edit 기능]
        |                                   └——> [Delete 기능]
        |
        ▼
    [필터 버튼 그룹] ——> [목록 필터링 (전체, 활성, 완료)]
```

- 코드 작성 및 주석:

  중요한 DOM 조작 또는 복잡한 이벤트 처리 부분에는 적절한 주석을 추가하여, 코드의 의도와 작동 과정을 명확하게 기록합니다.

기대 효과 및 심화 토론

이 과제를 통해 학생들은 단순히 static HTML/CSS에 머무르지 않고, 사용자와 상호작용하는 동적 웹 페이지를 만드는 법을 익힐 수 있습니다. JavaScript의 기본 메서드들을 풍부하게 사용하며, 이벤트 처리, DOM 조작, 애플리케이션 상태 관리 등 웹 프로그래밍의 핵심 개념을 경험하게 됩니다. 또한, 보너스 기능을 구현하는 과정에서 실제 제품 개발 시 고려해야 할 데이터 영속성이나 UX 개선 등 현실적인 문제도 다룰 수 있습니다.