📖 Todo List

이 문서는 제공된 Todo List의 JavaScript 코드를 단계별로 분석하여, 데이터 구조 설계부터 복잡한 UI 상호작용 로직까지 자바스크립트의 핵심 원리를 깊이 있게 이해할 수 있도록 돕습니다.

📌 기본 개념: 이 앱은 어떻게 작동하는가?

이 Todo List 애플리케이션은 데이터와 **UI (화면)**를 분리하여 관리하는 방식으로 작동합니다.

1. 데이터 저장: 모든 Todo 항목과 카테고리는 전역 변수로 정의된 JavaScript **배열(Array)**에 객체(Object) 형태로 저장됩니다.
2. UI 업데이트: 데이터가 변경될 때마다 (할 일 추가, 삭제, 완료 등) 배열을 수정하고, sortAndShowList() 함수를 호출하여 전체 목록을 다시 그려 화면을 갱신합니다.

Step 1. 데이터 모델 설계 및 정의

가장 먼저, 애플리케이션의 상태를 저장하는 핵심 데이터 구조를 이해해야 합니다.

1-1. 카테고리 **(categories)**

변수	설명	구조
categories	사용자가 정의한 카테고리 목록과 색상 코드를 저장합니다.	[{ name: "업무", color: "#3b82f6" }, ...]

1-2. 할 일 목록 **(todoList)**

속성	설명	예시
id	항목을 고유하게 식별하는 번호 (lastNo로 관리됨).	1
title	할 일 내용.	"JavaScript 복습"
done	완료 상태 (토글 기능의 핵심).	true 또는 false
category	할 일에 할당된 카테고리 이름.	"학습"

color	해당 카테고리의 색상 (렌더링에 사용).	"#f59e0b"
dueDate	마감일 (정렬 및 기한 초과 확인에 사용).	"2025-11-10"

1-3. 고유 ID 관리 (lastNo)

let lastNo = todoList.length > 0 ? Math.max(...todoList.map(item => item.id)) : 0;

- 역할: 새로운 Todo가 추가될 때마다 고유한 id를 부여하기 위해 현재 배열에서 가장 큰 ID를 찾아 저장합니다.
- 핵심 문법: Math.max(...array)를 사용하여 배열 내 객체의 특정 속성(id) 중 최댓값을 효율적으로 찾습니다.

Step 2. 화면 출력 및 렌더링 로직

데이터 객체를 실제 눈으로 볼 수 있는 HTML 요소로 변환하고, 데이터 변경 시 화면을 갱신하는 방법을 알아봅니다.

2-1. 단일 항목 생성 (getTodoItemElem(item))

이 함수는 Todo 객체 하나를 받아 완전한 <li> HTML 요소로 만듭니다.

1. document.createElement('li'): 새 리스트 항목을 생성합니다.
2. liElem.style.borderLeftColor = item.color;: CSS 스타일을 JavaScript로 직접 적용합니다.
3. 이벤트 리스너 부착:
   - titleElem.addEventListener('click', () => toggleDone(item.id));: 제목 클릭 시 완료 상태를 변경하는 함수(toggleDone)를 연결합니다.
   - deleteElem.addEventListener('click', ...): 삭제 버튼 클릭 시 removeItem 함수를 연결합니다.
4. 기한 초과(Overdue) 확인: item.dueDate < today 비교를 통해 마감일을 지났는지 확인하고, 기한 초과 시 텍스트 색상을 빨간색으로 변경하여 시각적 경고를 줍니다.

2-2. 전체 목록 갱신 (sortAndShowList()) - 바닐라 JS의 렌더링 핵심

이 함수는 데이터가 바뀔 때마다 실행되어 UI를 데이터와 일치시킵니다.

1. 데이터 정렬: [...todoList].sort(...)를 사용하여 현재 선택된 기준(마감일, 카테고리 등)에 따라 목록을 정렬합니다.
   - 주의: a.done ? 1 : -1 로직을 통해 완료된 항목을 항상 미완료 항목 뒤로 이동시켜 정렬합니다.
2. 화면 비우기: todoListUl.innerHTML = '';를 사용하여 기존에 그려져 있던 목록을 전부 지웁니다.
3. 목록 다시 그리기: 정렬된 sortedList 배열을 순회하며 각 항목에 대해

getTodoItemElem()을 호출하고, 반환된 <li> 요소를 ul에 순서대로 추가합니다.

Step 3. 핵심 CRUD 로직 (추가, 삭제, 완료)

데이터를 조작하는 가장 기본적인 함수들입니다.

함수	역할	핵심 로직
addItem()	새 할 일을 추가합니다.	lastNo 증가 후 새 객체를 만들어 todoList.push(newItem)
removeItem(id)	할 일을 삭제합니다.	todoList = todoList.filter(item => item.id !== id); (배열에서 해당 ID 제외)
toggleDone(id)	완료 상태를 변경합니다.	item.done = !item.done; (불리언 값을 반전)

❗ 모든 조작 후의 필수 동작: 위의 세 함수는 모두 데이터(배열)를 변경한 후, 반드시 sortAndShowList()를 호출하여 변경된 데이터를 화면에 반영해야 합니다.

Step 4. 카테고리 관리 로직 (심화)

단순한 Todo 목록을 넘어, 종속적인 데이터를 관리하는 방법을 보여줍니다.

4-1. 카테고리 추가 (addCategory())

이 함수는 중복 체크(categories.some(...)) 후 새 카테고리를 categories 배열에 추가하고, 다음 두 함수를 호출하여 화면을 갱신합니다.

- populateCategoryList(): 모달 창에 카테고리 칩 목록을 다시 그립니다.
- populateCategories(): Todo 입력 폼의 <select> 드롭다운 옵션을 다시 채웁니다.

4-2. 카테고리 삭제 (removeCategory(name))

이것은 로직이 가장 복잡한 부분 중 하나입니다.

1. 기본 카테고리 보호: "미지정" 카테고리는 삭제할 수 없도록 보호합니다.
2. 카테고리 제거: categories = categories.filter(c => c.name !== name);를 사용하여 카테고리를 배열에서 제거합니다.
3. 데이터 마이그레이션 (중요): 삭제된 카테고리를 사용하던 모든 Todo 항목을 찾아서(todoList.forEach) 기본 카테고리인 "미지정"으로 카테고리와 색상을 강제 업데이트합니다.
4. UI 갱신: populateCategoryList(), populateCategories(), sortAndShowList()를 모두

호출하여 카테고리 목록과 Todo 목록 전체를 갱신합니다.

Step 5. 디버깅 및 사용자 피드백

5-1. 콘솔 로깅 **(console.log())**의 역할

이전에 콘솔에 아무것도 뜨지 않았던 문제를 해결하기 위해, 데이터가 변경되는 모든 중요 지점(추가, 삭제, 토글, 카테고리 관리)에 console.log()가 추가되었습니다.

- 목적**:** 개발자가 앱 내부의 **배열 데이터(todoList, categories)**가 실제로 어떻게 실시간으로 변하고 있는지 눈으로 확인하고 디버깅할 수 있도록 돕습니다.

5-2. 알림창 **(showNotification())**의 역할

- 목적**:** 브라우저 기본 경고창(alert())을 사용하지 않고, 사용자에게 더 부드럽고 시각적으로 피드백(성공, 경고, 오류 메시지)을 제공하기 위해 커스텀 알림창을 구현한 함수입니다. CSS를 사용하여 3초 후 사라지도록 애니메이션 처리됩니다.

🚀 다음 단계로 나아가기 위한 학습 주제

이 앱은 바닐라 JS로 강력한 기능을 구현했지만, 실제 서비스를 위해서는 몇 가지 한계를 극복해야 합니다.

1. 데이터 영속성 **(Persistence)**

현재 데이터는 브라우저를 닫으면 사라집니다. 데이터를 영구적으로 저장하려면 다음을 학습해야 합니다.

- **Local Storage:** 브라우저에 간단한 문자열 데이터를 저장하는 기술. 데이터를 저장할 때 JSON.stringify()를, 불러올 때 JSON.parse()를 사용해야 합니다.
- **Firebase / Firestore:** 실시간 데이터베이스를 활용하여 여러 기기 간에 데이터를 동기화하는 백엔드 기술.

2. 컴포넌트 기반 아키텍처

sortAndShowList() 함수가 전체 목록을 매번 지우고 다시 그리는 것은 비효율적입니다. 복잡한 앱을 만들 때는 데이터 변경된 부분만 빠르게 업데이트하는 기술이 필요합니다.

- 학습 주제**:** React, Vue, Angular 같은 프론트엔드 프레임워크의 작동 원리 (Virtual DOM, 컴포넌트 개념)를 학습하면 이 문제를 근본적으로 해결할 수 있습니다.

3. UI/UX 개선

- 모듈화된 **CSS:** 현재는 모든 스타일이 <style> 태그 안에 있습니다. Tailwind CSS 또는 CSS Modules를 사용하여 스타일을 체계적으로 관리하는 방법을 익힐 수 있습니다.
- 접근성 **(Accessibility):** 키보드 탐색이나 스크린 리더 사용자를 위해 aria-* 속성을 추가하여 접근성을 높이는 방법을 학습해야 합니다.