8253jang@daum.net
장희정

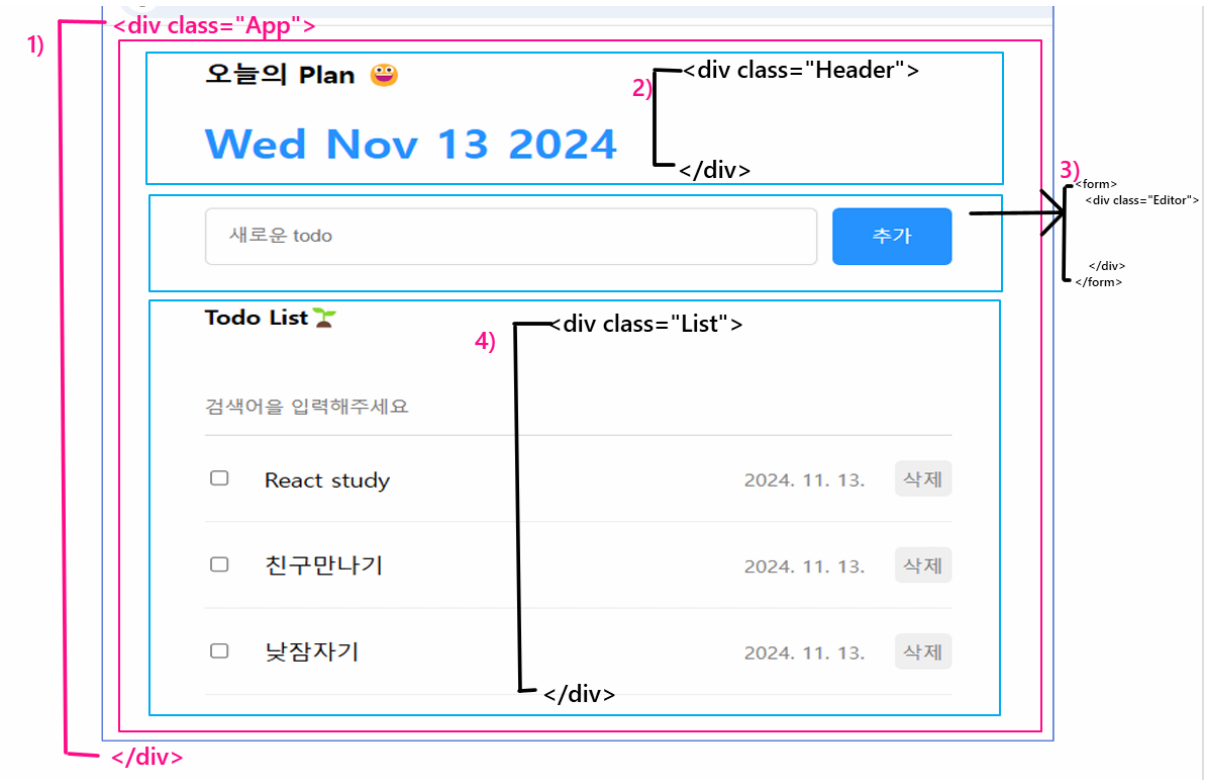
Todo_List Project (HTML, CSS , JavaScript)

화면구성(HTML + CSS)

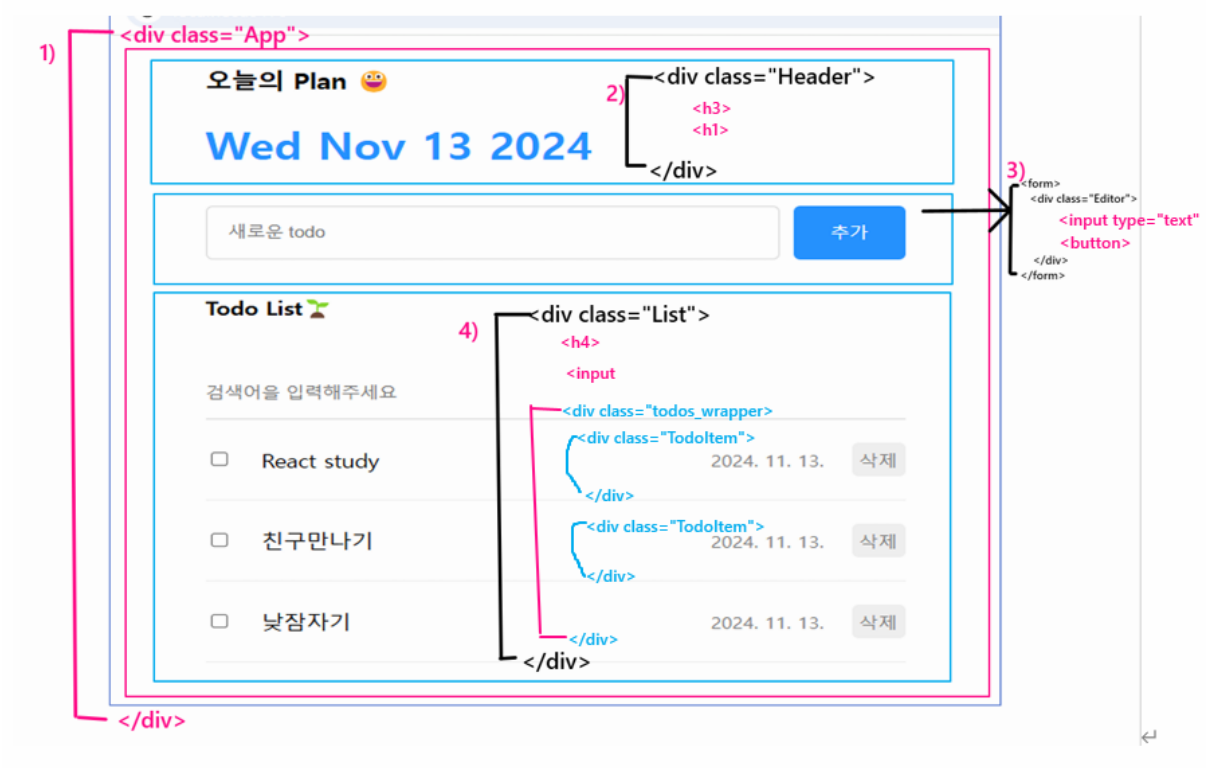
주요기능

1)전체출력
2)추가(등록)
3)수정(checkbox 상태변경)
4)삭제
5)검색

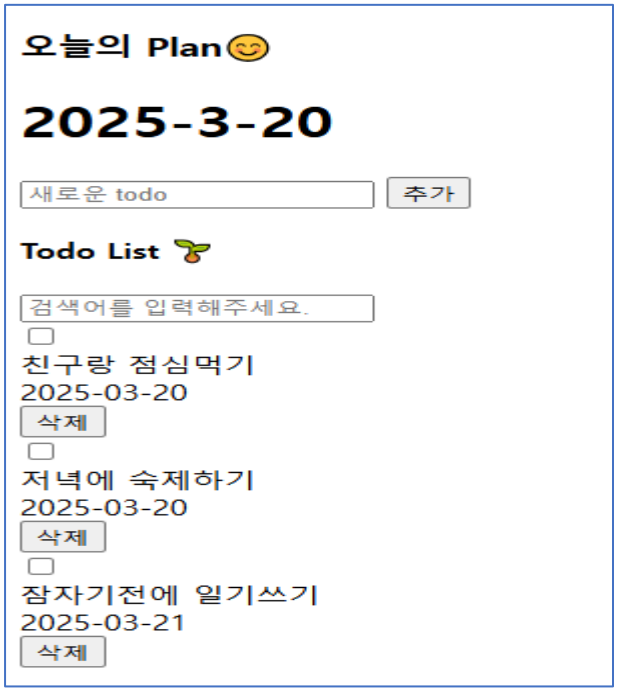
전체 UI



8253jang@daum.net
장희정



Step01 - HTML(ex01_TodoList_UI.html)



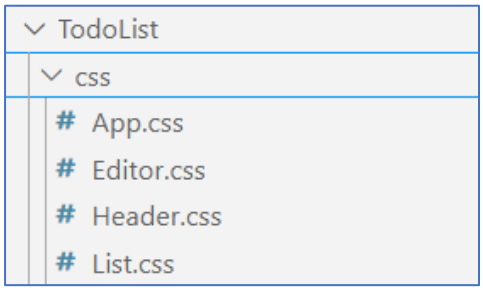
Window key + . 을 누르면 "이모지" 아이콘 나온다.

8253jang@daum.net
장희정

```
<div class="App">
  <div class="Header">
    <h3>
    <h1>
  </div>

  <form>
  <div class="Editor">
    <input
    <button>
  </div>
 </form>

  <div class="List">
    <h4>
    <input   />
    <div class="todos_wrapper">
      <div class="TodoItem">
        <input  />
        <div class="content">   </div>
        <div class="date">  </div>
        <button onClick="">  </button>
      </div>
      <div class="TodoItem">
        <input  />
        <div class="content">   </div>
        <div class="date">  </div>
        <button onClick="">  </button>
      </div>
      <div class="TodoItem">
        <input  />
        <div class="content">   </div>
        <div class="date">  </div>
        <button onClick="">  </button>
      </div>
    </div>
  </div>
</div>
```

8253jang@daum.net
장희정

Step02 - CSS적용(ex02_TodoList_CSS.html)

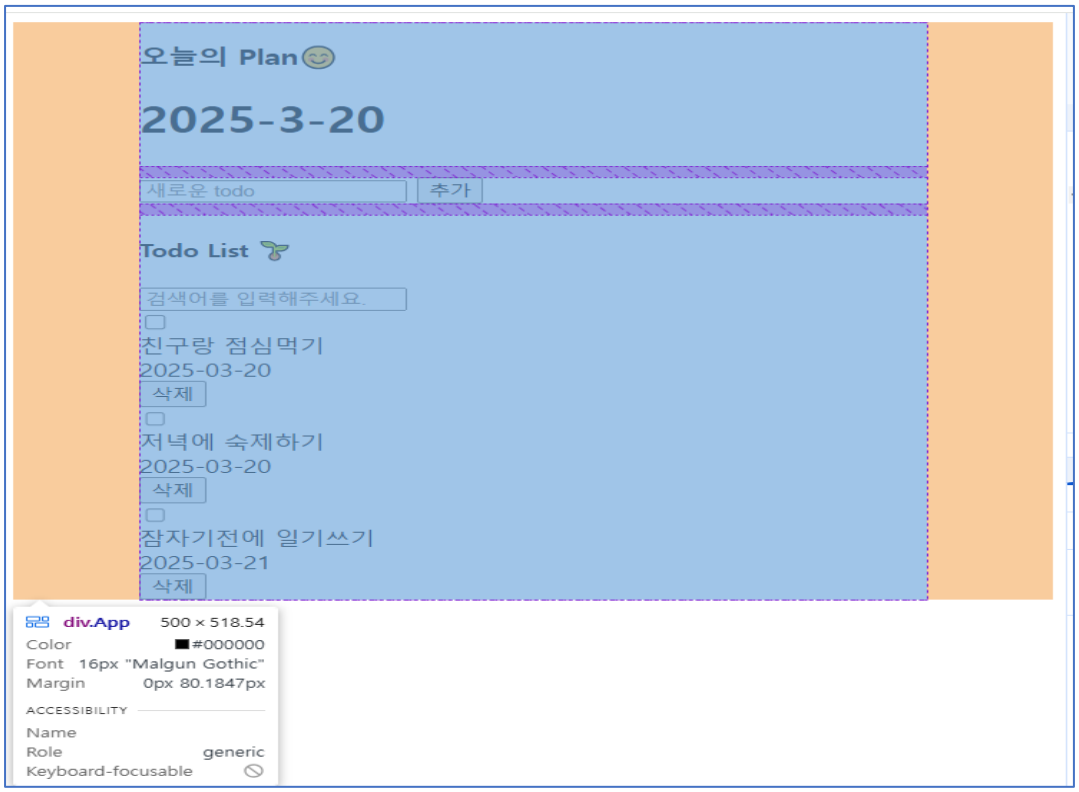


css > App.css

아래의 사항을 적용하세요

```
/* class App 요소
   1)display속성을 flex 적용
   2)방향은 가로방향 기준
   3)element사이의 갭은 10px
   4)가로 크기를 500px
   5)상하  좌우 margin은 0 auto
*/
```

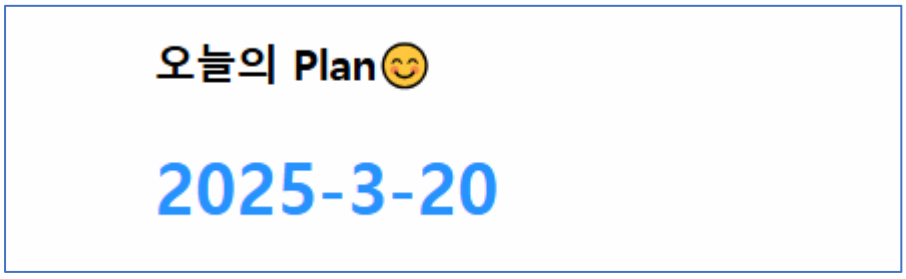
8253jang@daum.net
장희정



css > Header.css

아래의 사항을 적용하세요

```
/* 클래스 Header의 하위 자식 h1 요소
    1) color의 속성을  rgb(37, 147, 255) 로 적용
*/
```

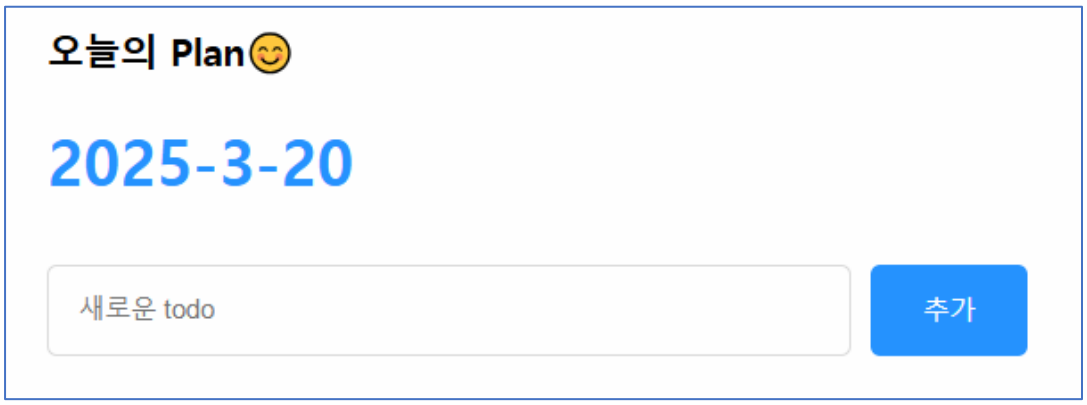


8253jang@daum.net
장희정

css > Editor.css

아래의 사항을 적용하세요

```
/*   class Editor에
   1)   display 속성 flex
   2)   gap 10px
*/

/* class Editor의 하위 input요소
   1)flex : 1   적용하여 부모요소에 벗어나지 않는 범위내에서 최대 크기로 나오게한다.
   2)padding 15px 적용
   3)외곽 테두리 1px solid rgb(220,220,220); 적용
   4)테두리를 둥글게 5px 적용
 */

   /* class Editor의 하위 button요소
   1)cusor 을 pointer로 설정
   2)가로크기 80px
   3)외곽테투리 none설정
   4)글자색상 white
   5)바탕색은 rgb(37,147,255)
   6)테두리를 둥글게 5px 적용
*/
```



8253jang@daum.net
장희정

css > List.css

아래의 사항을 적용하세요

```
/* class List 에
 1)display 속성 flex,
 2)flex-direction은 column 설정
 3)gap은 20px
*/

/*
  class List의 하위 자식 input요소
  1)가로크기 100%
  2)외곽선 none
  3)외곽의 아래(bottom) 부분 1px solid rgb(220, 220, 220) 설정
  4)패딩 상하 15px 좌우 0px 설정
*/

/*
 class List의 하위 자식 input:focus 요소
  1) outline : none설정(요소의 윤곽선 없애기)
  2) 외곽의 아래(bottom) 부분 1px solid rgb(37, 147, 255); 설정
*/

/*
  class List의 하위 자식 class todos_wrapper 요소
   1)display 속성 flex
   2)lex-direction은 column
   3)gap은 20px
*/
```

8253jang@daum.net
장희정

css > TodoItem.css

아래의 사항을 적용하세요

```
/* class TodoItem 요소
   1)display 속성 flex,
   2)align-items를 center (열을 기준으로 가운데 정렬)
   3)gap은 20px
   4)아래쪽 padding은 20px
   5)아래쪽 외곽선은  1px solid rgb(240, 240, 240);
*/

/*
 class TodoItem 하위 자식 input 요소
   1)  가로크기 20px;
*/

/*
 class TodoItem 하위 자식 class content 요소
   1) flex : 1 설정( 1은 부모요소에 벗어나지 않는 범위내에서 최대 크기로 나온다. )
;
```

```
*/

/*
 class TodoItem 하위 자식 class date 요소
   1)글자 크기 14px
   2)글자색상 gray
*/

/*
 class TodoItem 하위 자식 class button 요소
   1)cursor : pointer 설정
   2)글자색상 gray
   3)글자크기 14px
   4)외곽선 none
   5)테두리 둥글게 5px
   6) padding은 5px
*/
```

Step03 - JavaScript적용(ex03_TodoList_JS.html)

1)전체출력
2)추가(등록)
3)수정(checkbox 상태변경)
4)삭제
5)검색

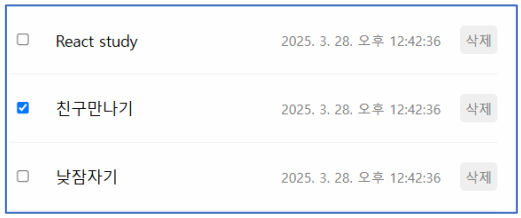
8253jang@daum.net
장희정

전체출력 기능

```
//초기 데이터
let mockData = [
        {id:0, isDone:false, content:"React study", date: new Date().getTime()},
        {id:1, isDone:true, content:"친구만나기", date: new Date().getTime()},
        {id:2, isDone:false, content:"낮잠자기", date: new Date().getTime()},
        ];

// 요일 출력을 위한 배열
let day =["일","월","화","수","목","금","토"];
```

기능

```
<script>
  onload = ()=>{
      //initData(mokData) 함수 호출

      // 현재 날짜를 년 월 일 요일로 출력한다.
            2025년 3월 28일 금요일
  }


  const initData = (printData)=>{
    // mockData 배열을 forEach를 이용해서  화면에 출력한다.
```




```
        /* checkbox
              onchange="onUpdate(id값)"
              todo의 isDone이 true이면 checked속성 추가

           button
             name속성 추가해서  value에 todo의 id 설정
             onclick ="todoDel(this)" 추가
```

```
        */
    }
```

추가 기능

```
let idIndex= 3; // id의 값을 증가 시킬 변수(초기데이터가 2까지 있으므로 3부터 시작)
document.querySelector(".Editor > button").onclick =() =>{
        event.preventDefault(); //전송기능 막음

       //id는 idIndex,
        isDone은 기본 false,
        content는 입력한 내용,
        date는 new Date().getTime()

        준비된 하나의 레코드를 mokData에 push()함수를 이용해서 추가한다.

        initData(mockData); //호출한다.(다시 화면 랜더링)
}
```

수정 기능

```
const onUpdate = (targetId)=>{ //TodoItem에서 호출할 때 전달한 id
    /* mockData의 state의 값들 중에 targetId와 일치하는 todoitem의 isDone 변경
        map함수를 이용한다. map함수의 결과를 mockData에 저장한다.
     */


    initData(mockData); //호출한다.(다시 화면 랜더링)
}
```

삭제 기능

```
const todoDel = (th)=>{
    //filter()함수를 이용해서 삭제하려는 대상이외의 todo만 추출해서 mockData에 담든다.
```

```
    initData(mockData); //호출한다.(다시 화면 랜더링)
}
```

검색 기능

```
document.querySelector("#keyword").onkeyup = (e)=>{

        let searchedTodos =     getFilterData(e.target.value);

        initData(searchedTodos);

    }


    const getFilterData = (search) =>{
          //검색어가 없으면 mockData를 리턴한다.
           if(search===""){
              return   mockData;
          }

          //filter함수를 이용해서 search(검색어)를 포함하고 있는 todo들을 받는다

         //filter의 결과를 리턴 한다.
    }
```