TodoList Application

TodoList Management

Todo class

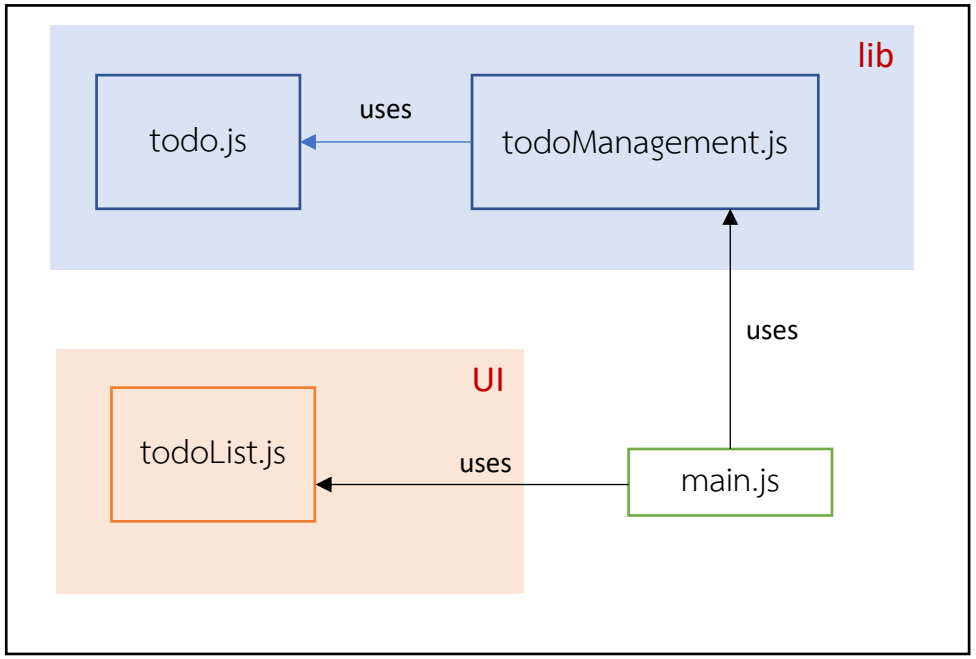
```
static runningId = 1
//properties
-id //unique id
-description //todo detail


//methods
//returns {id, description} object
-getTodo()
//allows to edit description
-setDescription(newDescription)
```

TodoList Management function

```
- todos //stores all todo in array
//adds a new todo to the end of todos array, returns the new length of the array
- function addTodo(desc)
//returns a todo object that its id equals to searchId
- function findTodo(searchId)
//returns an index of todo that its id equals to searchId
- function findIndexTodo(searchId)
//removes a todo that its id equals to removeId
- functon removeTodo(removeId)
//returns todos array
- function getTodos()
```

Document Object Modeling

(DOM)
TodoList Project Diagram



1. ไฟล์ ./lib/todo.js เพื่อจัดการกับ property และฟังก์ชันของแต่ละ todo
2. ไฟล์ ./lib/todoManagement.js เพื่อจัดการโครงสร้างข้อมูลและฟังก์ชันของ todo ทั้งหมด
3. ไฟล์ ./UI/todoListUI.js เพื่อจัดการ DOM ของเอกสาร HTML
4. ไฟล์ ./main.js สำหรับทดสอบไฟล์ต่าง ๆ ที่เกี่ยวข้อง

index.html

```
<html lang="en">

<head>
    <title>Todo List Application</title>
    <style>
      #listTodo {
        border-style: solid;
        border-width: 1pt;
        border-color: gray;
        padding: 10pt;
      }
      .todoItem {
        display: flex;
        align-items: center;
      }
      button {
        margin-right: 5pt;
        height: 25pt;
      }
      p {
        margin-right: 8pt;
      }
      input {
        height: 20pt;
      }
    </style>
  </head>
```

```
  <body>
    <div>
      <div id="addTodo">
        <h2>Add Your Todo:</h2>
        <input placeholder="Add Todo" />
        <button id="addBtn">Add</button>
      </div>
      <h1>### Your Todo list ###</h1>
      <div id="listTodo"></div>
      <div id="summaryTodo">
        <p id="done">Number of Done:</p>
        <p id="notDone">Number of Not Done:</p>
      </div>
    </div>
    <script src="./main.js" type="module"></script>
  </body>
</html>
```

ปรับปรุงไฟล์ ./lib/todo.js เพื่อจัดการกับ property และฟังก์ชันของแต่ละ todo

1. ให้เพิ่ม property done (Boolean) และปรับ constructor ให้รับ id, description, done เป็นพารามิเตอร์ เพื่อสร้าง todo object โดยถ้าค่า id เป็น undefined ให้ค่า id มีค่าเป็น running Id สำหรับ done ให้มีค่า default เป็น false
2. ให้เพิ่ม property done และเพิ่ม setDone method เพื่อกำหนดค่าให้กับ done

ปรับปรุงไฟล์ ./lib/todoManagement.js เพื่อจัดการโครงสร้างข้อมูลและฟังก์ชันของ todo ทั้งหมด

1. ให้แก้ไขฟังก์ชัน addTodo(desc) ให้ return todo.id ของรายการ todo ที่เพิ่มแทนการ return length ของ array
2. ให้เพิ่มฟังก์ชัน getNumberOfDone เพื่อ return จำนวนของ Todo ที่อยู่ในสถานะ Done
3. ให้เพิ่มฟังก์ชัน getNumberOfNotDone เพื่อ return จำนวนของ Todo ที่อยู่ในสถานะ Not Done
4. ให้เพิ่มฟังก์ชัน clearTodo เพื่อ reset ค่า todo ให้เป็น empty array

สร้างไฟล์ ./UI/todoListUI.js เพื่อจัดการ dom ของเอกสาร HTML

1.ให้เพิ่มฟังก์ชันที่ชื่อ showTodoItem(*newId, newDescription*) โดยทำการรับค่า todo id และ description เป็นพารามิเตอร์ เพื่อทำการแสดงรายการ todo ภายใต้ `<div id="listTodo"></div>` ของเอกสาร HTML โดยแต่ละรายการ todo ให้มีโครงสร้าง tags เรียงลำดับ ดังนี้

```
<div class="todoItem" id="newId">
        <p>newDescription</p>
        <button>Not done</button>
        <button>remove</button>
</div>
```

2. ให้เพิ่มฟังก์ชัน showNumberOfDone(numberOfDone) เพื่อแสดงเลขจำนวนของ Todo ที่อยู่ในสถานะ Done ต่อท้ายข้อความภายใน `<p id="done">Number of Done:</p>`
3. ให้เพิ่มฟังก์ชัน showNumberOfNotDone(numberOfNotDone) เพื่อแสดงเลขจำนวนของ Todo ที่อยู่ในสถานะ Not Done ต่อท้ายข้อความ ภายใน `<p id="notDone">Number of Not Done:</p>`

สร้างไฟล์ ./main.js

1. ให้ทดสอบเรียกใช้ addTodo() และรับ id ที่ return จากฟังก์ชัน เพื่อส่งต่อให้ฟังก์ชัน showTodoItem() เพื่อทำการสร้าง tags แสดงผลใน HTML File
2. ให้ทดสอบเรียกใช้ getNumberOfDone() และ getNumberOfNotDone() และรับค่าจำนวนของ Todo ที่อยู่ในสถานะ Done และ NotDone เพื่อส่งต่อฟังก์ชัน showNumberOfDone() และ showNumberOfNotDone() เพื่อแสดงผลใน HTML File