## Detailed Documentation for the Uzakov's To-Do List Code

# English

## Detailed Command Descriptions

### 1. `addTodo(event)`

- **Purpose**: Adds a new task to the to-do list.
- **Parameters**:
  - `event`: Prevents the default behavior of the form submission.
- **Steps**:
  1. **Create a** `div`: A container for the task.
     - Adds a class `todo` for styling.
  2. **Create a** `li`: The task itself.
     - Fills the `li` with the text from the input field (`todoInput.value`).
     - Adds a class `todo-item` for styling.
  3. **Save the task to local storage** using `saveLocalTodos(todoInput.value)`.
  4. **Add buttons**:
     - **Complete Button**: A button to mark the task as completed.
       - Includes an icon (`<i class="fas fa-check-circle"></i>`).
       - Adds a class `complete-btn`.
     - **Trash Button**: A button to delete the task.
       - Includes an icon (`<i class="fas fa-trash"></i>`).
       - Adds a class `trash-btn`.
  5. **Append the task and buttons** to the main container (`todoList`).
  6. **Clear the input field** for new entries.

## 2. `deleteCheck(e)`

- **Purpose**: Handles deletion and completion of tasks.
- **Parameters**:
  - `e`: Event object to identify the clicked element.
- **Steps**:
  - **Deletion**:
    1. Check if the clicked element has the class `trash-btn`.
    2. Add a transition animation to the task (`todo.classList.add("slide")`).
    3. Remove the task from local storage using `removeLocalTodos(todo)`.
    4. Wait for the transition to complete, then remove the task (`todo.remove()`).
  - **Completion**:
    1. Check if the clicked element has the class `complete-btn`.
    2. Toggle the class `completed` on the task (`todo.classList.toggle("completed")`), marking it as complete/incomplete.

---

## 3. `filterTodo(e)`

- **Purpose**: Filters tasks based on their status (all, completed, incomplete).
- **Parameters**:
  - `e`: Event object to capture the selected filter value.
- **Steps**:
  1. Retrieve all tasks (`todoList.childNodes`).
  2. Loop through each task and determine its visibility:
     - **"all"**: Show all tasks (`todo.style.display = "flex"`).
     - **"completed"**: Show only tasks with the `completed` class.
     - **"incomplete"**: Show only tasks without the `completed` class.

---

## 4. `saveLocalTodos(todo)`

- **Purpose**: Saves tasks to local storage.
- **Parameters**:
  - `todo`: The task text to save.
- **Steps**:
  1. Check if `todos` already exists in local storage:
     - If not, initialize an empty array.
  2. Add the new task to the array.
  3. Update the local storage with the new array (`localStorage.setItem`).

---

## 5. `getLocalTodos()`

- **Purpose**: Loads tasks from local storage when the page is reloaded.
- **Steps**:
  1. Check if `todos` exists in local storage:
     - If not, initialize an empty array.
  2. Loop through each saved task:
     - Re-create the `div`, `li`, and buttons as in `addTodo`.
     - Append them to `todoList`.

---

## 6. `removeLocalTodos(todo)`

- **Purpose**: Deletes a specific task from local storage.
- **Parameters**:
  - `todo`: The `div` containing the task to delete.
- **Steps**:
  1. Retrieve the saved tasks from local storage.
  2. Find the task in the array by matching its text (`todo.children[0].innerText`).

3. Remove the task from the array.
4. Update local storage with the modified array.

---

# Русский

## Подробное описание команд

### 1. `addTodo(event)`

- **Назначение**: Добавляет новую задачу в список дел.
- **Параметры**:
  - `event`: Предотвращает стандартное поведение отправки формы.
- **Этапы**:
  1. **Создать** `div`: Контейнер для задачи.
     - Добавляет класс `todo` для стилизации.
  2. **Создать** `li`: Элемент задачи.
     - Заполняет `li` текстом из поля ввода (`todoInput.value`).
     - Добавляет класс `todo-item` для стилизации.
  3. **Сохранить задачу в локальное хранилище** с помощью `saveLocalTodos(todoInput.value)`.
  4. **Добавить кнопки**:
     - **Кнопка завершения**: Кнопка для отметки задачи как завершённой.
       - Содержит иконку (`<i class="fas fa-check-circle"></i>`).
       - Добавляет класс `complete-btn`.
     - **Кнопка удаления**: Кнопка для удаления задачи.
       - Содержит иконку (`<i class="fas fa-trash"></i>`).
       - Добавляет класс `trash-btn`.
  5. **Добавить задачу и кнопки** в основной контейнер (`todoList`).

6. **Очистить поле ввода** для новых задач.

---

## 2. `deleteCheck(e)`

- **Назначение**: Обрабатывает удаление и завершение задач.
- **Параметры**:
  - `e`: Объект события для идентификации нажатого элемента.
- **Этапы**:
  - **Удаление**:
    1. Проверить, имеет ли нажатый элемент класс `trash-btn`.
    2. Добавить задаче анимацию перехода (`todo.classList.add("slide")`).
    3. Удалить задачу из локального хранилища с помощью `removeLocalTodos(todo)`.
    4. После завершения анимации удалить задачу (`todo.remove()`).
  - **Завершение**:
    1. Проверить, имеет ли нажатый элемент класс `complete-btn`.
    2. Переключить класс `completed` на задаче (`todo.classList.toggle("completed")`), отмечая её как завершённую/незавершённую.

---

## 3. `filterTodo(e)`

- **Назначение**: Фильтрует задачи по их статусу (все, завершённые, незавершённые).
- **Параметры**:
  - `e`: Объект события для получения значения фильтра.
- **Этапы**:
  1. Получить все задачи (`todoList.childNodes`).
  2. Для каждой задачи определить её видимость:

- **"all"**: Показать все задачи (`todo.style.display = "flex"`).
- **"completed"**: Показать только задачи с классом `completed`.
- **"incomplete"**: Показать только задачи без класса `completed`.

---

## 4. `saveLocalTodos(todo)`

- **Назначение**: Сохраняет задачи в локальное хранилище.
- **Параметры**:
  - `todo`: Текст задачи для сохранения.
- **Этапы**:
  1. Проверить, существует ли `todos` в локальном хранилище:
     - Если нет, инициализировать пустой массив.
  2. Добавить новую задачу в массив.
  3. Обновить локальное хранилище новым массивом (`localStorage.setItem`).

---

## 5. `getLocalTodos()`

- **Назначение**: Загружает задачи из локального хранилища при перезагрузке страницы.
- **Этапы**:
  1. Проверить, существует ли `todos` в локальном хранилище:
     - Если нет, инициализировать пустой массив.
  2. Для каждой сохранённой задачи:
     - Воссоздать `div`, `li` и кнопки, как в `addTodo`.
     - Добавить их в `todoList`.

---

## 6. `removeLocalTodos(todo)`

- **Назначение**: Удаляет конкретную задачу из локального хранилища.
- **Параметры**:

    - `todo`: `div`, содержащий задачу для удаления.
- **Этапы**:
    1. Получить сохранённые задачи из локального хранилища.
    2. Найти задачу в массиве по её тексту (`todo.children[0].innerText`).
    3. Удалить задачу из массива.
    4. Обновить локальное хранилище модифицированным массивом.

---