Лабораторная работа №3: Создание To-Do листа

Цель работы:

Изучить основные принципы разработки React-приложений с использованием инструментов Vite и Yarn через создание простого To-Do листа. В ходе работы студенты познакомятся с архитектурой React-компонентов, обработкой состояний (state), взаимодействием с пользователем через события.

---

1. Введение

React: основы и принципы

React — это библиотека JavaScript, основная задача которой заключается в создании пользовательских интерфейсов (UI). React основан на компонентном подходе, где каждый компонент может представлять собой отдельную часть страницы (например, форма или кнопка). Важные аспекты React:

- **Компоненты**: строительные блоки React-приложений.
- **JSX**: синтаксис, который позволяет писать HTML-подобные структуры прямо в JavaScript.
- **Состояние (State)**: динамические данные, которые могут изменяться со временем.
- **События (Events)**: позволяют пользователю взаимодействовать с компонентами.
- **Props**: это параметры, которые передаются в компонент для его настройки.

Vite: краткий обзор

Vite — это современный инструмент сборки, который ориентирован на высокую скорость разработки. Основные преимущества Vite:

- **Мгновенная перезагрузка модулей**.
- **Минимальные настройки**. Для начала работы не нужно писать много конфигураций.
- **Поддержка ES-модулей**. Это упрощает структуру проекта и ускоряет разработку.

Yarn: управление зависимостями

Yarn — это альтернативный менеджер пакетов к NPM, обеспечивающий более высокую производительность, безопасность и стабильность. Преимущества использования Yarn:

- **Кэширование зависимостей**.
- **Быстрая установка** благодаря параллельной загрузке.
- **Консистентные версии пакетов** благодаря файлу yarn.lock.

---

2. Подготовка окружения

1. Скачайте и установите **Node.js** с официального сайта nodejs.org. Это позволит вам использовать Node.js для разработки на JavaScript.
2. Проверьте версию Node.js в терминале:

   ```
   node -v
   ```

3. Установите **Yarn**:

   ```
   npm install --global yarn
   ```

4. Проверьте установку Yarn:

   ```
   yarn --version
   ```

---

3. Создание проекта

1. Для создания нового React-проекта с использованием Vite выполните следующую команду:

```
yarn create vite todo-app --template react
```

2. Перейдите в директорию проекта:

```
cd todo-app
```

3. Установите все необходимые зависимости:

```
yarn
```

4. Запустите проект для разработки:

```
yarn dev
```

5. Откройте браузер по указанному адресу (обычно это `http://localhost:5173`).

---

4. Структура проекта

В Vite-проекте структура папок будет примерно такой:

```
todo-app/
├── public/
├── src/
│   ├── App.jsx
│   ├── main.jsx
│   └── index.css
├── package.json
└── yarn.lock
```

- **src/** — папка с исходным кодом приложения.
- **App.jsx** — главный компонент приложения.

- **main.jsx** — точка входа для всего приложения.

---

5. Создание To-Do листа

1. **Структура компонентов**

   В To-Do листе будет несколько ключевых компонентов:

   - **App** — главный компонент, содержащий всю логику.
   - **TodoItem** — отдельная задача (to-do).
   - **TodoList** — список задач.

2. **Создание компонента** `TodoItem`

Создайте компонент для одной задачи:

```jsx
// src/components/TodoItem.jsx
import React from 'react';

function TodoItem({ todo, toggleTodo }) {
  return (
    <div>
      <input
        type="checkbox"
        checked={todo.completed}
        onChange={() => toggleTodo(todo.id)}
      />
      <span>{todo.title}</span>
    </div>
  );
}

export default TodoItem;
```

1. **Создание компонента** `TodoList`

Теперь создадим компонент для отображения списка задач:

```jsx
// src/components/TodoList.jsx
import React from 'react';
import TodoItem from './TodoItem';

function TodoList({ todos, toggleTodo }) {
  return (
    <div>
      {todos.map((todo) => (
        <TodoItem key={todo.id} todo={todo} toggleTodo={toggl
eTodo} />
      ))}
    </div>
  );
}

export default TodoList;
```

1. **Главный компонент** `App`

Теперь нужно объединить всё в главном компоненте:

```jsx
// src/App.jsx
import React, { useState } from 'react';
import TodoList from './components/TodoList';

function App() {
  const [todos, setTodos] = useState([
    { id: 1, title: 'Buy groceries', completed: false },
    { id: 2, title: 'Read a book', completed: false },
  ]);
  const [newTodo, setNewTodo] = useState('');

  const toggleTodo = (id) => {
    setTodos(
      todos.map((todo) =>
```

```
        todo.id === id ? { ...todo, completed: !todo.complete
d } : todo
      )
    );
  };

  const addTodo = (e) => {
    e.preventDefault();
    if (!newTodo) return;

    const newTodoItem = {
      id: Date.now(),
      title: newTodo,
      completed: false,
    };
    setTodos([...todos, newTodoItem]);
    setNewTodo('');
  };

  return (
    <div>
      <h1>My To-Do List</h1>
      <form onSubmit={addTodo}>
        <input
          type="text"
          value={newTodo}
          onChange={(e) => setNewTodo(e.target.value)}
          placeholder="Add new task..."
        />
        <button type="submit">Add</button>
      </form>
      <TodoList todos={todos} toggleTodo={toggleTodo} />
    </div>
  );
}
```

```
export default App;
```

6. Задания для выполнения

1. **Добавьте возможность удаления задач.**

   Для этого вам нужно реализовать функционал удаления задачи. Каждая задача должна иметь кнопку удаления, при нажатии на которую она удаляется из списка.

2. **Реализуйте фильтрацию задач по состоянию (выполненные/ невыполненные).**

   Добавьте функционал фильтрации, который позволит отображать все задачи, только выполненные или только невыполненные. Для этого можно добавить кнопки или выпадающий список, которые будут переключать отображение списка задач.

3. **Стилизация приложения.**

   Добавьте базовые стили для улучшения внешнего вида To-Do листа. Вы можете использовать обычный CSS, а также любую библиотеку стилей (например, **TailwindCSS**).