Лабораторная работа № 2_3
«Работа с HTML и CSS в проекте React + Vite + TypeScript»

Цель: освоить создание React-проекта на базе Vite и TypeScript, научиться работать с HTML-элементами и стилями в компонентах.

Шаг 1. Создание проекта

1. Откройте терминал в VS Code (Ctrl + `` ` `` или View → Terminal).
2. Выполните команды:

```
npm create vite@latest lab-2_3_surname-- --template react-ts
cd lab-react-css
npm install
```

3. Запустите проект: `npm run dev`.
4. Откройте в браузере ссылку из вывода терминала (обычно http://localhost:5173).

Шаг 2. Очистка проекта

В папке `src/` удалите:

- `App.css`
- `index.css`
- `main.tsx` (оставьте только `main.jsx`)
- `vite-env.d.ts` (если есть)

Шаг 3. Создание App.tsx

Создайте файл src/App.tsx.

Вставьте код:

```tsx
import React from 'react';

function App(): JSX.Element {
  return (
    <div>
      <h1>Привет!</h1>
    </div>
  );
}

export default App;
```

Шаг 4. Импорт App.tsx в main.tsx

Откройте main.tsx.

Замените содержимое на:

```tsx
import React from 'react';
import ReactDOM from 'react-dom/client';
import App from './App';

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
);
```

Сохраните файл. Проверьте браузер — должен отображаться заголовок «Привет!».

Шаг 5. Работа с createElement

В App.tsx закомментируйте функцию App и JSX-разметку.

Ниже добавьте реализацию через React.createElement:

```
// const App: React.FC = () => {
//   return <div><h1>Привет!</h1></div>;
// };

const App = (): JSX.Element => {
  return React.createElement('div', null,
    React.createElement('h1', null, 'Привет!')
  );
};
```

Сохраните. Проверьте браузер — результат должен быть идентичен.

Шаг 6. Возврат к JSX и добавление тегов

1.Раскомментируйте исходный вариант с JSX (шаг 3).

2.Дополните разметку несколькими разными тегами:

```jsx
function App(): JSX.Element {
  return (
    <div className="app-container">
      <h1>Привет!</h1>
      <p>Это лабораторная работа по React + Vite + TypeScript.</p>
      <ul>
        <li>Пункт списка 1</li>
        <li>Пункт списка 2</li>
        <li>Пункт списка 3</li>
      </ul>
      <button type="button">Нажми меня</button>
    </div>
  );
}
```

3.Сохраните файл. Проверьте браузер — все элементы должны отображаться корректно.

Шаг 7. Создание стилевого файла для App.tsx

Создайте файл src/App.module.css (использование CSS Modules обеспечит локальную область видимости стилей).

Добавьте несколько CSS-правил:

```css
.app-container {
  text-align: center;
  padding: 20px;
  background-color: #f5f5f5;
  font-family: Arial, sans-serif;
}

.app-container h1 {
  color: #333;
  margin-bottom: 10px;
}

.app-container p {
  color: #666;
  line-height: 1.5;
}

.app-container ul {
  list-style-type: none;
  padding: 0;
}
```

Шаг 8. Импорт и подключение стилей

Вернитесь в файл src/App.tsx.

Добавьте импорт CSS-модуля в начало файла:

```
import styles from './App.module.css';
```

Обновите JSX-разметку, применив классы из CSS-модуля:

```
function App(): JSX.Element {
  return (
    <div className={styles['app-container']}>
      <h1 className={styles['app-container-h1']}>Привет!</h1>
      <p className={styles['app-container-p']}>Это лабораторная работа по React + Vite + TypeScript.</p>
      <ul className={styles['app-container-ul']}>
        <li className={styles['app-container-li']}>Пункт списка 1</li>
        <li className={styles['app-container-li']}>Пункт списка 2</li>
        <li className={styles['app-container-li']}>Пункт списка 3</li>
      </ul>
      ... (продолжение Вашего кода)

```

Примечание: для упрощения можно использовать более короткий синтаксис, если классы переименовать в appContainer, h1 и т. д.:

```
<div className={styles.appContainer}>
  <h1 className={styles.h1}>Привет!</h1>
  {/* ... */}
</div>
```

Дополнительные задания :

- Добавьте ещё один компонент Header.tsx, вынесите туда заголовок и подключите в App.tsx.
- Создайте отдельный файл стилей для Header.tsx.
- Попробуйте добавить интерактивность (например, при нажатии на кнопку меняется текст).