Лабораторная работа № 2_1
"Знакомство с Vite + React"

Инструменты сборки Vite

Цель работы:

Познакомиться с современным инструментом сборки `Vite`, научиться создавать проект, запускать его в режиме реального времени и изучать структуру файлов.

Необходимые инструменты:

- `Node.js` (версия 18+)
- `npm` / `yarn` / `pnpm`
- Терминал (командная строка)

Теория

Vite (фр. "быстрый") — это инструмент сборки front-end проектов.

Почему он быстрый?

1. **Мгновенный запуск сервера:** Вместо сборки всего проекта, Vite использует нативные ES-модули в браузере.
2. **Быстрая горячая замена модулей (HMR):** При изменении кода обновляется только измененный модуль, а не вся страница.

Преимущества для разработки:

- Простота настройки (ноль конфигурации для начала).
- Поддержка TypeScript, JSX, CSS из коробки.
- Оптимизированная сборка для продакшена.

Шаг 1. Создание проекта

1. В VSCode откройте общую папку для всех заданий по дисциплине (например, “web”).
2. Откройте окно терминала и перейдите в общую папку, если требуется:

```
cd..
```

3. Проверка установленного ПО:

```
node -v,  npm -v
```

4. Если *node.js* не установлен, установите:

https://nodejs.org/en/

Для установки *node.js* также можно использовать команду: `winget install OpenJS.NodeJS.LTS`

5. Выполните команду для создания нового проекта с шаблоном *react* с помощью утилиты *Vite* (вместо *surname* вписать Вашу фамилию латиницей):

Команда для создания:

```
# Создаем проект с именем my-first-vite-app
# Шаблон react включает поддержку JavaScript и React
npm create vite@latest lab2_1_surname -- --template react
```

Шаг 2. Установка

После запуска команды потребуется установка **create-vite@8.3.0** – согласитесь с установкой (**Y**).
Щелкните [Enter] несколько раз в знак согласия.
Стрелкой вниз на клавиатуре переместитесь на пункт *React*:

```
◆ Select a framework:
  ○ Vanilla
  ○ Vue
  ● React
  ○ Preact
  ○ Lit
  ...
```

Затем:

```
◆ Select a variant:
  ○ TypeScript
  ○ TypeScript + React Compiler
  ○ TypeScript + SWC
  ● JavaScript
  ○ JavaScript + React Compiler
  ...
```

Далее:

```
◆ Use Vite 8 beta (Experimental)?:
  ○ Yes
  ● No
```

И:

```
◆ Install with npm and start now?
  ● Yes / ○ No
```

Откройте браузер. Перейдите по адресу, который выдал терминал: http://localhost:5173/ . Вы должны увидеть стартовую страницу Vite + React. Это означает, что сервер успешно запущен. Приложение работает!

Шаг 3. Первый взгляд в браузере

1. Откройте браузер.
2. Перейдите по адресу, который выдал терминал: `http://localhost:5173/`

Вы должны увидеть стартовую страницу `Vite + React`.

- Это означает, что сервер успешно запущен.
- Приложение работает!

Шаг 4. Режим "Live Reload" (Горячая перезагрузка)

Оставим терминал и браузер открытыми рядом. Сейчас мы увидим главную "магию" современных фреймворков — **Hot Module Replacement (HMR)**.

Задача:
Найти файл `src/App.jsx` (или `App.js`) в вашем редакторе кода и изменить текст.

Было:

```
// Было (приблизительно строка 10)
<h1>Vite + React</h1>
```

Стало:

```
// Меняем текст приветствия
<h1>Моё первое приложение на Vite!</h1>
```

Шаг 5. Наблюдаем за результатом

Как только вы сохраните файл (`Ctrl+S` или `Cmd+S`), посмотрите на браузер.

Ожидаемый результат:
Страница **не перезагрузится целиком**, а просто изменит текст заголовка на новый. Состояние счетчика (если вы его нажимали) должно сохраниться.

- Это и есть работа **HMR (Hot Module Replacement)**.
- Vite отправил в браузер только измененный кусок кода (модуль `App.jsx`).

Перезапуск сервера

Если браузер не обновился – выполните команды терминала:

Команда для запуска:

```
ctrl + C
npm run dev
или
pnpm run dev
```

Шаг 6. Осмотр структуры проекта

Давайте разберемся, из чего состоит наше приложение. Откройте папку проекта в редакторе кода (VS Code).

Ключевые файлы и папки:

```
my-first-vite-app/
├── node_modules/      # Папка с библиотеками (не трогаем руками)
├── public/            # Статические файлы (favicon.svg и т.д.)
├── src/               # Исходный код нашего приложения (работаем здесь!)
│   ├── assets/        # Картинки, стили, шрифты
│   ├── App.jsx        # Главный компонент приложения
│   ├── App.css        # Стили для главного компонента
│   ├── main.jsx       # ТОЧКА ВХОДА — рендерит App в DOM
│   └── index.css      # Глобальные стили
├── index.html         # ГЛАВНЫЙ HTML-файл (входная точка для браузера)
├── package.json       # Список зависимостей и скриптов
├── vite.config.js     # Файл конфигурации Vite
└── README.md          # Документация проекта
```

Ключевой файл — index.html

В отличие от многих других сборщиков (`Create React App`), в `Vite` файл `index.html` находится не в `public`, а **в корне проекта**.

Фрагмент кода `index.html`:

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/vite.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Vite + React</title>
  </head>
  <body>
    <!-- Корневой элемент, куда React будет рендерить приложение -->
    <div id="root"></div>
    <!-- Точка входа JavaScript (подключается модульно) -->
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Точка входа JavaScript — main.jsx

Файл `src/main.jsx` отвечает за "склеивание" React-компонента с реальным DOM-деревом браузера.

Фрагмент кода `src/main.jsx`:

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App.jsx' // Импортируем наш главный компонент
import './index.css'       // Импортируем глобальные стили

// Находим div с id="root" из index.html
ReactDOM.createRoot(document.getElementById('root')).render(
  <React.StrictMode>
    { /* Рендерим наше приложение внутрь root */ }
    <App />
  </React.StrictMode>,
)
```

Шаг 7. css

Измените цвет фона в файле *src/App.css*, добавив его для селектора root.

Шаг 8. Создание компонента-кнопки

Создайте новый компонент *MyButton.jsx* в папке *src* и импортируйте его в *App*.jsx:

- Создайте файл MyButton.jsx. Это можно сделать:
  - через командную строку (если установлено расширение): touch MyButton.jsx;
  - через IDE (например, VS Code) — просто щёлкните правой кнопкой мыши в папке *src* → *«New File»* → назовите файл *MyButton.jsx*.
- Откройте *MyButton.jsx* в редакторе и добавьте код компонента:

```
1   function MyButton() {
2       return (
3
4           <div>
5               <button> Моя кнопка!
6               </button>
7           </div>
8
9       )
10  }
11
12  export default MyButton
```

- Импортируйте компонент в *App.jsx*:

```jsx
import MyButton from './MyButton';

function App() {
  const [count, setCount] = useState(0)

  return (
    <>
      <div>
        <a href="https://vite.dev" target="_blank">
          <img src={viteLogo} className="logo" alt="Vite logo" />
        </a>
        <a href="https://react.dev" target="_blank">
          <img src={reactLogo} className="logo react" alt="React logo" />
        </a>
      </div>
      <h1>Моё первое приложение!</h1>
      <MyButton>Нажми меня!</MyButton>
```

Шаг 9. Удаление ненужного

Удалите все ненужные файлы и команды импорта, логотипы, оставив только вывод заголовка и кнопки:

- Очистить полностью *index.css*.
- В *App. css* оставить правила только для селектора *root*.
- Удалить логотипы из папок *assets* и *public*.
- В *App.jsx* удалить ненужные импорты и теги:

```jsx
import { useState } from 'react'
import './App.css'
import MyButton from './MyButton';

function App() {
  return (
    <>
      <h1>Моё первое приложение!</h1>
      <MyButton>Нажми меня!</MyButton>
    </>
  )
}
export default App
```

- Посмотрите результат в браузере.

Шаг 10. Создание production-версии

1. Остановите сервер и соберите production-версию командой:

```
ctrl+C
```

```
npm run build
```

Появилась папка *dist*

2. Запустите локальный сервер для проверки (например, с http-server):

```
npx http-server dist
```

На вопрос установить ли сервер, выберите (**Y**)

3. Откройте сайт в браузере по указанному в терминале адресу и проверьте работо способность страницы: http://127.0.0.1:8080/

Дополнительное задание:

1. Создать новый компонент `MyButtonClick.jsx` в папке `src` и импортировать его в `App.jsx`. Кнопка должна при щелчке выводить окно с сообщением «Кнопка нажата».
2. Остановить сервер (`Ctrl+C` в терминале) и собрать production-версию командой `npm run build`. Посмотреть, какая папка появилась.

Вопросы для самопроверки:

- За что отвечает папка `public`?
- Что произойдет, если удалить тег `<script type="module" src="/src/main.jsx">` из `index.html`?

Выполнение

Создайте файл компонента MyButtonClick.jsx.

Добавьте безымянную функцию с двумя параметрами:

```jsx
const MyButton = ({ children, onClick }) => {
  return (
    <button onClick={onClick} className="my-button">
      {children}
    </button>
  );
};

export default MyButton;
```

{children} — позволяет передавать текст или элементы внутрь кнопки;

onClick — обработчик клика (можно передавать из родительского компонента);

Выполнение

Добавьте импорт компонента в начало файла App.jsx (вместе с другими импортами):

```
import MyButtonClick from './MyButtonClick';
```

Добавьте компонент MyButtonClick в JSX-разметку внутри App:

```
function App() {
  const handleClick = () => {
  alert('Кнопка нажата!');
 };
 return (
  <div className="App">
   <h1>Моё первое приложение!</h1>
   <MyButton>Нажми меня!</MyButton>
   <MyButtonClick onClick={handleClick}>Нажми меня!</MyButtonClick>
  </div>

 )
}
export default App;
```

handleClick — функция, которая выполнится при клике по кнопке;
текст "Нажми меня!" станет содержимым кнопки (children).

Выводы

Что мы сегодня сделали:

1. Сгенерировали проект с помощью `Vite`.
2. Разобрались с основными командами (`create`, `install`, `run dev`).
3. Убедились в скорости `Hot Module Replacement (HMR)`.
4. Изучили структуру современных React-приложений.

Vite — это мощный и простой инструмент, который ускоряет разработку. Полученная структура является стандартом для большинства современных front-end проектов.