# Настройка окружения для веб-разработки через терминал VS Code (Windows)
# Лабораторная работа 1.2

**Цель работы:** Научиться устанавливать и настраивать необходимое программное обеспечение для веб-разработки с использованием терминала Visual Studio Code, понимать назначение каждого инструмента и уметь проверять корректность установки.
В этой работе мы установим все необходимые компоненты, по возможности используя **терминал**, а там, где это невозможно — через официальные установщики.

**Почему терминал?** Умение работать с командной строкой — ключевой навык разработчика. Это быстрее, автоматизируемо и часто является единственным способом установки в серверных средах (Linux без графического интерфейса).

# Подготовительный этап: Установка и настройка VS Code

**Шаг 0.1. Скачивание и установка Visual Studio Code**

1. Если VS Code ещё не установлен:
2. Перейдите на официальный сайт: https://code.visualstudio.com/
3. Скачайте установщик для Windows (System Installer рекомендуется).
4. Запустите скачанный файл и выполните установку, оставляя все параметры по умолчанию.
5. **Важно:** На этапе "Выбор дополнительных задач" отметьте галочкой **"Добавить действие 'Открыть с помощью Code' в контекстное меню файлов"** и **"Добавить в PATH"**.

**Шаг 0.2. Открытие терминала в VS Code**

1. Запустите Visual Studio Code.
2. Откройте встроенный терминал: нажмите Ctrl + `` (обратный апостроф) или выберите в меню **Terminal → New Terminal**.
3. Убедитесь, что в правом нижнем углу терминала указан **PowerShell** или **Command Prompt**. Если там bash (из-за WSL), переключитесь на PowerShell, нажав на выпадающий список.

# Модуль 1: Установка Node.js (Среда выполнения JavaScript)

**Зачем нужен Node.js?**

**Node.js** — это среда выполнения JavaScript вне браузера. Она позволяет запускать JavaScript-код на сервере, создавать серверные приложения, инструменты сборки и утилиты командной строки. Без Node.js невозможна работа современных фронтенд-инструментов (Vite, Webpack) и пакетных менеджеров.

**Установка через терминал (рекомендуемый способ)**

1. В терминале VS Code (PowerShell) выполните команду для установки fnm:

   powershell

   **winget install Schniz.fnm**

*Примечание:* Если winget не работает, установите fnm вручную по инструкции.

1. Перезапустите терминал или выполните команду для применения изменений:

   powershell

   **fnm env --use-on-cd | Out-String | Invoke-Expression**

2. Установите последнюю LTS-версию Node.js:

   powershell

   **fnm install --lts**

4. Настройте использование этой версии по умолчанию:

powershell

**fnm default lts-latest**

**5. Проверка установки:**

powershell

**node --version**

**npm --version**

Вы должны увидеть номера версий (например, v20.18.0 и 10.8.2).

**Альтернативный способ (если терминал не работает)**

Скачайте установщик с официального сайта: https://nodejs.org/ (выбирайте LTS-версию).

# Модуль 2: Установка pnpm (Пакетный менеджер)

**Зачем нужен pnpm?**

**pnpm** — это быстрый и эффективный менеджер пакетов для JavaScript/Node.js. В отличие от npm и yarn, он использует глобальное хранилище и жёсткие ссылки, что экономит место на диске и ускоряет установку зависимостей. Современные проекты всё чаще переходят на pnpm

**Установка через терминал (PowerShell)**

Самый простой способ установить pnpm — через встроенный менеджер пакетов Node.js (npm).

1. В терминале выполните команду:

```
npm install -g pnpm
```

Флаг -g означает глобальную установку (доступность во всей системе).

**2. Проверка установки:**

```
pnpm --version
```

Вы должны увидеть номер версии (например, 9.12.0).

**Альтернативный способ (скрипт)**

Если npm не работает, можно использовать отдельный скрипт:

```
iwr https://get.pnpm.io/install.ps1 -useb | iex
```

# Модуль 3: Установка Git (Система контроля версий)

**Зачем нужен Git?**

**Git** — это распределённая система контроля версий. Она позволяет отслеживать изменения в коде, возвращаться к предыдущим версиям, работать в команде (ветвление, слияние) и публиковать код на GitHub/GitLab

**Установка через терминал**

В Windows проще всего установить Git через официальный установщик, но можно и через winget:

1. Выполните команду:
   ```
   winget install --id Git.Git -e --source winget
   ```
2. После установки перезапустите терминал VS Code (иначе переменные PATH могут не обновиться).
3. **Проверка установки:**

   ```
   git --version
   ```

Вы должны увидеть git version 2.4x.x.

**Первоначальная настройка Git**

Обязательно укажите своё имя и email (они будут подписывать ваши коммиты):

```
git config --global user.name "Ваше Имя"
git config --global user.email "ваш-email@example.com"
```

**Альтернативный способ**

Скачайте установщик с официального сайта: https://git-scm.com/download/win

# Модуль 4: Установка WSL (Подсистема Windows для Linux)

**Зачем нужен WSL?**

**WSL (Windows Subsystem for Linux)** позволяет запускать полноценную среду Linux прямо в Windows без виртуальной машины. Это необходимо для разработки, так как многие серверные инструменты (Docker, некоторые базы данных) изначально созданы для Linux и лучше работают в нативной среде

**Установка через терминал (PowerShell с правами администратора) (пока не надо?)**

**Важно:** Терминал нужно запустить **от имени администратора**.

1. В терминале VS Code нажмите стрелку вниз рядом со значком "+" и выберите **"Command Prompt"** или убедитесь, что PowerShell открыт от имени администратора.
2. Выполните команду для установки WSL с дистрибутивом Ubuntu по умолчанию:

```
wsl --install
```

Эта команда включает необходимые компоненты Windows, скачивает ядро Linux и устанавливает Ubuntu.

3. После завершения установки **перезагрузите компьютер**, когда система попросит.
4. После перезагрузки откроется консоль с предложением создать имя пользователя и пароль для Linux. Запомните их!
5. В VS Code теперь можно выбрать WSL-терминал: нажмите на выпадающий список в терминале и выберите **"Ubuntu (WSL)"**.

**Проверка установки**

В терминале WSL выполните:

```
bash
    lsb_release -a
```

Вы должны увидеть информацию об Ubuntu.

# Модуль 5: Установка Docker (Контейнеризация)

**Зачем нужен Docker?**

**Docker** позволяет упаковывать приложения и их зависимости в контейнеры — изолированные среды, которые гарантированно работают одинаково на всех компьютерах (разработчика, тестировщика, сервере). Для веб-разработки Docker часто используется для запуска баз данных, очередей и других сервисов

Docker требует наличия WSL2. Поскольку мы уже установили WSL, установка Docker Desktop пройдёт гладко.

1. **Скачайте Docker Desktop** с официального сайта: https://www.docker.com/products/docker-desktop/ (Через терминал установить Docker Desktop невозможно, нужен графический установщик).
2. Запустите скачанный установщик и следуйте инструкциям.
3. **Важно:** В процессе установки убедитесь, что отмечена опция **"Use WSL 2 instead of Hyper-V"** (использовать WSL 2 вместо Hyper-V).
4. После установки перезагрузите компьютер.
5. Запустите Docker Desktop (он должен появиться в системном трее).

**Проверка установки**

Откройте терминал VS Code (можно PowerShell) и выполните:

```
powershell
docker --version
docker-compose --version
```

Также можно проверить интеграцию с WSL, выполнив в терминале Ubuntu:

```
bash
docker ps
```

(Должен отобразиться пустой список контейнеров, а не ошибка).

# Модуль 6: Установка PostgreSQL (Реляционная база данных)

**Зачем нужен PostgreSQL?**

**PostgreSQL** (Postgres) — одна из самых мощных и популярных реляционных баз данных с открытым исходным кодом. Она поддерживает сложные запросы, транзакции, JSON-поля и используется в большинстве современных веб-приложений

**Установка через терминал (WSL)**

Рекомендуется устанавливать PostgreSQL в среде WSL (Linux), так как это ближе к реальной серверной среде.

1. Откройте терминал WSL (Ubuntu) в VS Code.
2. Обновите списки пакетов и установите PostgreSQL:

```
bash
sudo apt update
sudo apt install postgresql postgresql-contrib
```

3. Проверьте статус сервера:

```
bash
sudo service postgresql status
```

Если не запущен, запустите:

```
bash
sudo service postgresql start
```

4. Переключитесь на пользователя postgres и войдите в консоль PostgreSQL для создания пароля:

```
bash
sudo -u postgres psql
```

В консоли PostgreSQL выполните:

```
sql
ALTER USER postgres PASSWORD 'your_password';
\q
```

**Альтернативный способ (Windows-версия)**

Если вы не используете WSL, можно скачать установщик для Windows:

1. Перейдите на официальный сайт: https://www.postgresql.org/download/windows/
2. Скачайте установщик (EnterpriseDB) и запустите его.
3. В процессе установки запомните пароль для суперпользователя postgres.

**Проверка подключения**

В терминале WSL:

```
bash
sudo -u postgres psql -c "\l"
```

Должен отобразиться список баз данных.

# Модуль 7: Установка DBeaver (Клиент для баз данных)

**Зачем нужен DBeaver?**

**DBeaver** — это универсальный клиент для работы с базами данных. Он поддерживает PostgreSQL, MySQL, SQLite и многие другие СУБД. Предоставляет графический интерфейс для просмотра данных, выполнения SQL-запросов и администрирования. Существует бесплатная **Community Edition**, которой достаточно для учебных целей.

**Установка через терминал**

DBeaver можно установить через пакетный менеджер winget.

1. В терминале PowerShell выполните:

   powershell

   winget install DBeaver.DBeaver

2. После установки найдите DBeaver в меню "Пуск" и запустите.

**Альтернативный способ (сайт)**

Скачайте Community Edition с официального сайта: https://dbeaver.io/download/ (выбирайте версию для Windows).

**Настройка для работы быстрее (по желанию)**

По умолчанию DBeaver скачивает драйверы из центрального репозитория Maven, что может быть медленно. Рекомендуется добавить зеркало Aliyun :

1. Откройте **Окно → Настройки** (Windows) или **DBeaver → Preferences** (Mac).
2. Перейдите в раздел **Подключения → Драйверы → Maven**.
3. Нажмите **Добавить**, введите URL: http://maven.aliyun.com/nexus/content/groups/public/ и нажмите OK.
4. Переместите эту запись наверх с помощью кнопки **Вверх**

# Модуль 8: Установка Oh My Zsh (Улучшение терминала в WSL)

**Зачем нужен Oh My Zsh?**

**Oh My Zsh** — это фреймворк для управления конфигурацией Zsh-оболочки (Z shell). Он добавляет красивые темы, плагины (автодополнение, подсветка синтаксиса) и делает работу в терминале приятнее и продуктивнее. Устанавливается только в Linux-среду (WSL)

**Установка в WSL (Ubuntu)**

1. Откройте терминал WSL (Ubuntu) в VS Code.
2. Сначала установите Zsh (если его нет):

```
bash
sudo apt update
sudo apt install zsh -y
```

3. Установите Oh My Zsh с помощью curl (или wget):

```
bash
sh -c "$(curl -fsSL https://raw.githubusercontent.com/ohmyzsh/ohmyzsh/master/tools/install.sh)"
```

Если curl не установлен: sudo apt install curl -y.

4. В процессе установки скрипт спросит, хотите ли вы сменить shell по умолчанию на Zsh. Нажмите **Y** (Yes).
5. **Перезапустите терминал WSL.** Теперь ваша оболочка должна выглядеть по-другому (обычно появляется тема "robbyrussell").

# Установка полезных плагинов (опционально)

Добавим плагины для подсветки синтаксиса и автодополнения:

bash

```
# Клонируем плагин подсветки синтаксиса
git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plu
gins/zsh-syntax-highlighting

# Клонируем плагин автодополнения
git clone https://github.com/zsh-users/zsh-autosuggestions ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/z
sh-autosuggestions
```

Теперь отредактируйте конфигурационный файл:

```
bash
nano ~/.zshrc
```

Найдите строку plugins=(git) и замените её на:

```
bash
plugins=(git zsh-syntax-highlighting zsh-autosuggestions)
```

Сохраните файл (Ctrl+O, затем Ctrl+X). Примените изменения:

```
bash
source ~/.zshrc
```

# Заключительный этап: Проверка и сдача работы

**Финальная проверка**

Создайте простой скрипт или выполните команды, чтобы убедиться, что всё работает.

1. **Node.js и pnpm:**
   powershell
   pnpm init

Должен создаться файл package.json.

2. **Git:**
   powershell
   git status
3. **Docker:**
   powershell
   docker run hello-world
4. **PostgreSQL:**

В WSL: sudo -u postgres psql -c "SELECT version();"

5. **DBeaver:** Запустите программу и попробуйте создать подключение к локальной PostgreSQL (host: localhost, порт: 5432, пользователь: postgres, пароль: ваш пароль).
6. **Oh My Zsh:** Просто посмотрите на красивый терминал и попробуйте набрать git + пробел + Tab — должно появиться автодополнение.