GitHub для студентов веб-программирования

Полное руководство по работе с Git и GitHub в рамках курса

Что такое GitHub и зачем он нужен?

GitHub - это платформа для:

- **Хранения кода** в облаке (не потеряется при поломке компьютера)
- **Отслеживания изменений** (можно откатиться к любой версии)
- **Совместной работы** над проектами
- **Демонстрации работ** работодателям (ваше портфолио)

В нашем курсе GitHub используется для:

- Получения заданий от преподавателя
- Сдачи домашних работ и проектов
- Ведения портфолио ваших проектов
- Изучения современных инструментов разработки

ЭТАП 1: Регистрация и первоначальная настройка

Создание аккаунта GitHub (5 минут)

1. **Перейдите на** github.com
2. **Нажмите "Sign up"**
3. **Заполните форму:**
   - **Username:** Выберите профессиональное имя (например: `ivan-petrov`, `maria-dev`, `alex-webdev`)
   - **Email:** Используйте постоянный email
   - **Password:** Надежный пароль
4. **Подтвердите email** через письмо
5. **Выберите Free план** (его достаточно для обучения)

💡 **Совет по username:**

- Избегайте цифр рождения и случайных символов
- Используйте имя или никнейм, который не стыдно показать работодателю

- Примеры хороших username: `alexander-smith`, `frontend-maria`, `js-developer`

Установка Git на компьютер

Windows:

1. Скачайте Git с git-scm.com
2. Запустите установщик
3. Используйте настройки по умолчанию (везде Next)
4. В конце выберите "Git Bash" как терминал

macOS:

```bash
# Установка через Homebrew (если есть)
brew install git

# Или скачать с git-scm.com
```

Linux (Ubuntu/Debian):

```bash
sudo apt update
sudo apt install git
```

Настройка Git (первый запуск)

Откройте терминал (Git Bash на Windows) и выполните:

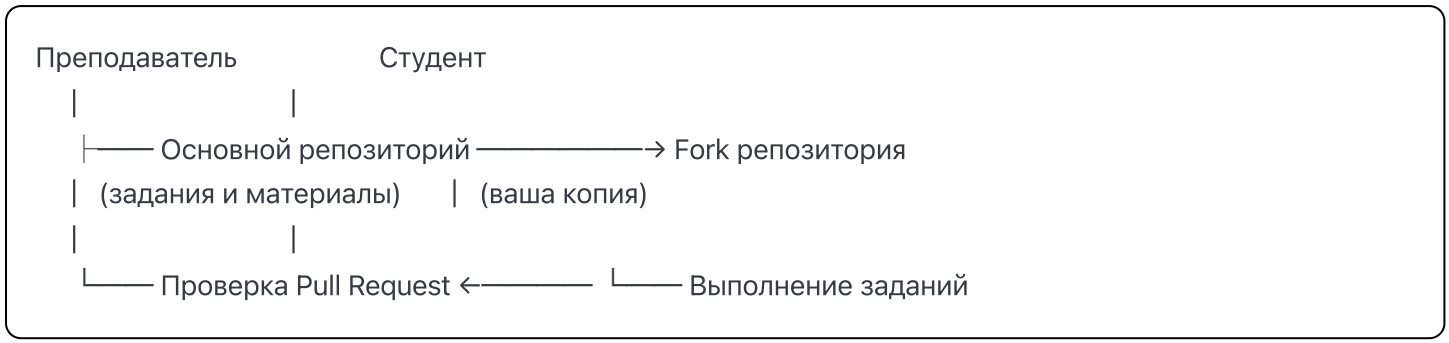
```bash
# Настройка имени пользователя
git config --global user.name "Ваше Имя"

# Настройка email (тот же, что и в GitHub)
git config --global user.email "your.email@example.com"

# Проверка настроек
git config --list
```

ЭТАП 2: Работа с репозиториями курса

Модель работы в нашем курсе:



Шаг 1: Форк репозитория курса

1. **Перейдите на страницу** основного репозитория курса
2. **Нажмите кнопку "Fork"** в правом верхнем углу
3. **Выберите свой аккаунт** как destination
4. **Дождитесь создания** форка (копии репозитория)

Теперь у вас есть собственная копия репозитория по адресу:
`https://github.com/YOUR_USERNAME/web-programming-course`

Шаг 2: Клонирование репозитория на компьютер

bash

```
# Создайте папку для проектов курса
mkdir ~/web-course
cd ~/web-course

# Клонируйте ваш форк (замените YOUR_USERNAME на ваш username)
git clone https://github.com/YOUR_USERNAME/web-programming-course.git

# Перейдите в папку проекта
cd web-programming-course

# Проверьте, что все скачалось
ls -la
```

Шаг 3: Настройка upstream (связь с основным репозиторием)

bash

```
# Добавляем основной репозиторий как upstream
git remote add upstream https://github.com/TEACHER_USERNAME/web-programming-course.git

# Проверяем настройку
git remote -v
```

Должно показать:

```
origin    https://github.com/YOUR_USERNAME/web-programming-course.git (fetch)
origin    https://github.com/YOUR_USERNAME/web-programming-course.git (push)
upstream  https://github.com/TEACHER_USERNAME/web-programming-course.git (fetch)
upstream  https://github.com/TEACHER_USERNAME/web-programming-course.git (push)
```

ЭТАП 3: Ежедневная работа с GitHub

Получение новых заданий от преподавателя

Перед началом каждого урока:

```bash
# Переходим в папку проекта
cd ~/web-course/web-programming-course

# Получаем изменения из основного репозитория
git fetch upstream

# Переключаемся на main ветку
git checkout main

# Применяем изменения из основного репозитория
git merge upstream/main

# Отправляем обновления в ваш форк
git push origin main
```

Выполнение домашнего задания

Создание новой ветки для задания:

```bash
```

```
# Создаем и переключаемся на новую ветку
git checkout -b homework-lesson-01

# Проверяем, что мы в правильной ветке
git branch
```

Структура папок для заданий:

```
web-programming-course/
├── lessons/
│   ├── lesson-01/
│   │   ├── homework/
│   │   │   ├── index.html
│   │   │   ├── style.css
│   │   │   └── script.js
│   │   └── materials/
│   ├── lesson-02/
│   └── ...
└── projects/
    ├── mini-landing/
    └── final-project/
```

Работа над заданием:

1. **Создайте папку для задания** (если не создана):

```
bash

mkdir -p lessons/lesson-01/homework
cd lessons/lesson-01/homework
```

2. **Создайте необходимые файлы** и выполните задание

3. **Регулярно сохраняйте прогресс:**

```
bash
```

```
# Добавляем все изменения в индекс
git add .

# Создаем коммит с описанием изменений
git commit -m "Добавлен базовый HTML для лендинга кофейни"

# Продолжаем работать...
# Еще изменения...

git add .
git commit -m "Добавлены стили и адаптивность"
```

Отправка выполненного задания

Шаг 1: Отправка ветки на GitHub

```
bash

# Отправляем ветку с заданием на GitHub
git push origin homework-lesson-01
```

Шаг 2: Создание Pull Request

1. **Перейдите в ваш форк** на GitHub
2. **GitHub покажет уведомление** о новой ветке с кнопкой "Compare & pull request"
3. **Нажмите "Compare & pull request"**
4. **Заполните форму Pull Request:**
   - **Title:** `Домашнее задание - Урок 1`
   - **Description:**

```
markdown
```

```
## Выполненное задание: Лендинг кофейни

### Что сделано:
- ✅ Создан HTML с семантической разметкой
- ✅ Добавлены стили CSS
- ✅ Реализована адаптивность для мобильных
- ✅ Добавлена простая интерактивность на JS

### Дополнительно:
- Добавил анимации при наведении
- Использовал CSS Grid для layout

### Вопросы:
- Правильно ли я использовал семантические теги?
- Можно ли улучшить производительность?

### Время выполнения: ~3 часа
```

5. **Нажмите "Create pull request"**

ЭТАП 4: Работа с обратной связью

Получение комментариев от преподавателя

Преподаватель может:

- **Оставить общие комментарии** к Pull Request
- **Прокомментировать конкретные строки** кода
- **Запросить изменения** (Request Changes)
- **Одобрить работу** (Approve)

Исправление замечаний

Если требуются доработки:

```bash
```

```
# Убедитесь, что вы в правильной ветке
git checkout homework-lesson-01

# Внесите необходимые изменения в код
# ...

# Сохраните изменения
git add .
git commit -m "Исправлены замечания преподавателя: улучшена семантика HTML"

# Отправьте обновления
git push origin homework-lesson-01
```

Pull Request автоматически обновится новыми коммитами.

После принятия работы

```
bash

# Возвращаемся на основную ветку
git checkout main

# Удаляем локальную ветку с заданием (она больше не нужна)
git branch -d homework-lesson-01

# Удаляем ветку на GitHub (опционально)
git push origin --delete homework-lesson-01
```

ЭТАП 5: Работа с крупными проектами

Создание отдельного репозитория для проекта

Для крупных проектов создаем отдельные репозитории:

1. **На GitHub нажмите "New repository"**
2. **Заполните информацию:**
   - **Repository name:** `coffee-shop-landing`
   - **Description:** `Лендинг кофейни - учебный проект курса веб-программирования`
   - **Public/Private:** Public (для портфолио)
   - **Add README:** ✅ Yes
   - **Add .gitignore:** Node (если будет использовать сборщики)
3. **Нажмите "Create repository"**

Клонирование и настройка проекта

bash

```bash
# Клонируем новый репозиторий
git clone https://github.com/YOUR_USERNAME/coffee-shop-landing.git
cd coffee-shop-landing

# Создаем базовую структуру проекта
mkdir src css js images
touch src/index.html src/css/style.css src/js/script.js

# Создаем первый коммит
git add .
git commit -m "Initial project structure"
git push origin main
```

Пример качественного README для проекта

markdown

☕ Coffee Shop Landing Page

Лендинг страница для кофейни, созданная в рамках курса "Веб-программирование с TypeScript".

🚀 Демо

[Посмотреть живой сайт](https://your-username.github.io/coffee-shop-landing)

🛠️ Технологии

- HTML5 (семантическая разметка)
- CSS3 (Flexbox, Grid, анимации)
- Vanilla JavaScript
- Responsive Design

📋 Функциональность

- [x] Адаптивный дизайн для всех устройств
- [x] Плавные анимации и переходы
- [x] Интерактивное меню
- [x] Форма обратной связи
- [x] Карта с локацией (Google Maps API)

🎨 Дизайн

Дизайн вдохновлен современными кофейнями с акцентом на:
- Теплые коричневые тона
- Минималистичный интерфейс
- Качественные изображения продукции
- Удобная навигация

📁 Структура проекта

```
coffee-shop-landing/
├── index.html
├── css/
│   ├── style.css
│   └── responsive.css
├── js/
│   ├── main.js
│   └── map.js
├── images/
│   └── ...
└── README.md
```

🚦 Установка и запуск

1. Клонируйте репозиторий:
```bash
git clone https://github.com/your-username/coffee-shop-landing.git
```

2. Откройте `index.html` в браузере или используйте Live Server

📚 Что изучено

В процессе создания проекта изучены:

- Семантическая HTML разметка
- CSS Grid и Flexbox для layout
- JavaScript для интерактивности
- Работа с API (Google Maps)
- Оптимизация изображений
- Git workflow

👤 Автор

Ваше Имя

- GitHub: @your-username
- Email: your.email@example.com

📝 Лицензия

Учебный проект - создано в образовательных целях.

ЭТАП 6: GitHub Pages - публикация проектов

Настройка GitHub Pages

1. **Перейдите в Settings** вашего репозитория
2. **Найдите раздел "Pages"** в левом меню
3. **В Source выберите:** "Deploy from a branch"
4. **В Branch выберите:** `main` и папку `/ (root)`
5. **Нажмите "Save"**

Через несколько минут ваш сайт будет доступен по адресу:
`https://YOUR_USERNAME.github.io/REPOSITORY_NAME`

Автоматическое деплой при изменениях

После настройки каждый `push` в main ветку автоматически обновит опубликованный сайт.

Полезные команды Git для ежедневной работы

Основные команды:
```bash
# Проверить статус репозитория
git status

# Посмотреть историю коммитов
git log --oneline

# Посмотреть изменения в файлах
git diff

# Отменить изменения в файле (до коммита)
git checkout -- filename.html

# Вернуться к предыдущему коммиту (осторожно!)
git reset --hard HEAD~1
```

Работа с ветками:

```bash
```

```
# Список всех веток
git branch -a

# Создать и переключиться на новую ветку
git checkout -b feature-new-section

# Переключиться между ветками
git checkout main
git checkout feature-new-section

# Удалить ветку (после merge)
git branch -d feature-new-section
```

Решение конфликтов:

```
bash

# Если возник конфликт при merge
git status  # покажет конфликтующие файлы

# Откройте файлы, исправьте конфликты
# (удалите маркеры <<<<<<< ======= >>>>>>>)

git add .
git commit -m "Resolved merge conflicts"
```

Частые проблемы и их решения

❌ "Permission denied" при push

✅ **Решение:** Настройте SSH ключи или используйте Personal Access Token:

```
bash

# Для HTTPS (временное решение)
git remote set-url origin https://USERNAME:TOKEN@github.com/USERNAME/REPO.git
```

❌ "Your branch is behind origin/main"

✅ **Решение:**

```
bash
```

```
git pull origin main
# или
git fetch origin
git merge origin/main
```

❌ Случайно закоммитили секретные данные

✅ **Решение:**

```
bash

# Удалить последний коммит (если еще не запушили)
git reset --soft HEAD~1

# Или создать новый коммит с исправлениями
git add .
git commit -m "Remove sensitive data"
```

❌ Забыли переключиться на новую ветку

✅ **Решение:**

```
bash

# Создать новую ветку из текущего состояния
git checkout -b forgotten-branch
```

Чек-лист для каждого задания

Перед началом работы:

- [ ] Получил последние изменения от преподавателя (`git pull upstream main`)
- [ ] Создал новую ветку для задания
- [ ] Понимаю требования к заданию

Во время работы:

- [ ] Делаю коммиты регулярно (каждые 30-60 минут работы)
- [ ] Пишу понятные сообщения коммитов
- [ ] Тестирую код перед коммитом

Перед отправкой:

- [ ] Код работает без ошибок
- [ ] Проверил на разных размерах экрана (если веб-страница)

- [ ] Создал подробное описание в Pull Request
- [ ] Указал время выполнения и возникшие сложности

После проверки:

- [ ] Исправил замечания преподавателя
- [ ] Поблагодарил за обратную связь
- [ ] Удалил ненужные ветки после принятия работы

Ресурсы для изучения Git

Интерактивные туториалы:

- Git Tutorial - Atlassian
- Learn Git Branching
- GitHub Skills

Полезные инструменты:

- **GitKraken** - графический интерфейс для Git
- **GitHub Desktop** - официальное приложение GitHub
- **VS Code Git integration** - встроенная поддержка Git

Документация:

- Official Git Documentation
- GitHub Guides

Помните: Git и GitHub - это навыки, которые пригодятся в любой IT-карьере. Уделите время изучению, и это окупится многократно!