

**Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University**

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 2

По дисциплине Web-программирование

Тема работы Основы Git и Galp

Обучающийся Алексеев Тимофей Юрьевич

Факультет Факультет инфокоммуникационных технологий

Группа K3221

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся	<u>01.10.2024</u> (дата)	<u> </u> (подпись)	<u>Алексеев Т.Ю.</u> (Ф.И.О.)
--------------------	-----------------------------	--	----------------------------------

Руководитель	<u> </u> (дата)	<u> </u> (подпись)	<u>Марченко Е.В.</u> (Ф.И.О.)
---------------------	---------------------------------------	--	----------------------------------

Санкт-Петербург
2024 г.

Цель

Познакомиться с основами работы с Git и Gulp.

Задачи

1. Ознакомиться с основными запросами Git;
2. Создать репозиторий, сделать 3 коммита и синхронизировать;
3. Установить gulp;
4. Создать простую задачу с помощью gulp;
5. Создать переключатель веб-страниц с определенным интервалом с помощью gulp.

Ход работы

Задание 1

В данном задании было необходимо произвести работу с системой контроля версий Git.

Для этого в заранее созданной ветке lab_2 был добавлен код из лабораторной работы 1. После чего в каждый из файлов были внесены небольшие изменения. Далее был сделан коммит с помощью команд git add – all и git commit -m “N commit”.

```
C:\Users\aleks\OneDrive\Рабочий стол\итмо\5 сем\WebDevelopment_2024-2025>git add --all
warning: in the working copy of 'works/K3321/Алексеев_Тимофей_Юрьевич/lab_2/code/ex1.html', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'works/K3321/Алексеев_Тимофей_Юрьевич/lab_2/code/ex2.html', LF will be replaced by CRLF
the next time Git touches it
warning: in the working copy of 'works/K3321/Алексеев_Тимофей_Юрьевич/lab_2/code/ex3.html', LF will be replaced by CRLF
the next time Git touches it
C:\Users\aleks\OneDrive\Рабочий стол\итмо\5 сем\WebDevelopment_2024-2025>git commit -m "Second commit"
[lab_2 6b6dcef] Second commit
3 files changed, 3 insertions(+), 3 deletions(-)
```

Рисунок 1 – Команды, с помощью которых создается коммит

Рефакторинг файлов был произведен два раза, в результате чего получилось сделать 3 коммита, которые можно посмотреть в истории репозитория.

Далее для синхронизации локального репозитория с удаленным была использована команда git push -u origin lab_2.

```
C:\Users\aleks\OneDrive\Рабочий стол\итмо\5 сем\WebDevelopment_2024-2025>git push -u origin lab_2
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 8 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (18/18), 1.40 KiB | 716.00 KiB/s, done.
Total 18 (delta 8), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (8/8), completed with 5 local objects.
To github.com:NorthPole0499/WebDevelopment_2024-2025.git
   e571443..925cf3d lab_2 -> lab_2
branch 'lab_2' set up to track 'origin/lab_2'.
```

Рисунок 2 – Синхронизация удаленного репозитория и локального

В итоге текущий проект доступен по следующей ссылке:
https://github.com/NorthPole0499/WebDevelopment_2024-2025/tree/lab_2

Задание 2

В данном упражнении требуется установить Gulp и создать task.

Для его установки пропишем в консоли следующие команды `npm install gulp npm` и `install gulp-cli`.

Далее проверим правильность установки путем запуска команды `npx gulp --version`. Получаем информацию об установленных версиях.

```
C:\Users\aleks\OneDrive\Рабочий стол\итмо\5 сем\WebDevelopment_2024-2025>npx gulp --version
CLI version: 3.0.0
Local version: 5.0.0
```

Рисунок 3 – Версия установленного gulp

Далее для создания первой задачи необходимо написать в консоль команду `npm init`, которая создает окружение.

```
{
  "name": "gulp_task",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}
```

Рисунок 4 – Информация о текущем окружении

После этого устанавливаем в наше окружение gulp с помощью команды `npm install gulp --save-dev`. Теперь мы готовы к созданию первой задачи.

Создаем файл `gulpfile.js` в корневой папке нашего проекта. После чего прописываем там следующий код, обозначающий инициализирование задачи под именем `hello`.

```
var gulp = require('gulp');

gulp.task('hello', function() {
  console.log('Hello world, this code was written by Timofey!');
});
```

Рисунок 5 – Код первой задачи Gulp

После этого добавляем в `package.json` в раздел `scripts` следующую команду под названием `hello`: `gulp hello`.

Прописав в консоли `npm run hello`, сможем увидеть результат выполнения первой задачи.

```
_task>npm run hello

> gulp_task@1.0.0 hello
> gulp hello

[20:18:06] Using gulpfile ~\OneDrive\Рабочий стол\lab_2\gulp_task\gulpfile.js
[20:18:06] Starting 'hello'...
Hello world, this code was written by Timofey!
```

Рисунок 6 – Выполнение задачи gulp

Задание 3

В данном задании было необходимо создать в `gulp` задачу, которая сама бы переключала страницы в браузере с определенным интервалом.

Для решения данной задачи нам понадобится установка веб-сервера с помощью команды `npm install browser-sync --save-dev`.

Далее нам необходимо создать папку `app`, в которой будут храниться все исполняемые файлы. А именно `index.html` и `script.js`.

В `index.html` мы прописываем стандартный синтаксис. В `head` мы укажем `css`-стиль для тега `iframe`, который растянет его на весь экран. В `body` же вызовем сам тег `iframe`, отображающий другие веб-страницы, а также укажем наш `script.js`.

```

<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Лабораторная 2</title>
  <style>
    iframe {
      width: 100%;
      height: 100vh;
      border: none;
    }
  </style>
</head>
<body>
  <iframe id="webFrame"></iframe>
  <script src="script.js"></script>
</body>
</html>

```

Рисунок 7 – Листинг кода файла index.html

После этого переходим к написанию script.js. В ней мы объявляем константы: адреса, интервал и тд. После этого создаем функцию ChangePage, которая присваивает атрибуту src у тега iframe адрес веб-страницы. После чего устанавливается вызов данной функции через указанный интервал.

```

const urls = [
  'https://my.itmo.ru/',
  'https://cap.ru/',
  'https://www.wikipedia.org'
];
const interval = 6000;
let currentIndex = 0;

function changePage() {
  const frame = document.getElementById('webFrame');
  frame.src = urls[currentIndex];
  currentIndex = (currentIndex + 1) % urls.length;
}

setInterval(changePage, interval);
changePage();

```

Рисунок 8 – Листинг кода файла script.js

После чего в файле gulpfiles.js создаем новую задачу, для которой используем наш веб-сервер с помощью browser-sync. При каждом изменении в каком-либо файле браузер будет перезагружаться.

```
gulp.task('opener', function () {  
  browserSync.init({  
    server: {  
      baseDir: './app'  
    }  
  });  
  
  gulp.watch('app/index.html').on('change', browserSync.reload);  
  gulp.watch('src/script.js').on('change', browserSync.reload);  
});
```

Рисунок 9 – Задача opener

В итоге заносим нашу задачу в package.json под именем open: gulp opener. После этого прописываем в консоль npm run open и наблюдаем, как в открывшейся вкладке переключаются страницы.

Вывод

В ходе выполнения лабораторной работы цель была достигнута. Было произведено ознакомление с основами работы с Git и Gulp.

В течение выполнения работы были изучены основные команды Git, создан репозиторий, сделаны коммиты и произведена синхронизация удаленного и локального репозитория. Установлен gulp и написана первая задача, выводящая приветственное сообщение. Написана задача, которая вызывает веб-сервер, который переключает сайты из списка с определенным интервалом.