

Лабораторная работа №1

Работа с git

Гнатюк Анастасия Станиславовна

Содержание

Цель работы	2
Теоретическое введение	2
Выполнение лабораторной работы	2
1.1 Подготовка	2
1.1.1 Установка имени и электронной почты	2
1.1.2 Параметры установки окончаний строк	2
1.1.3 Установка отображения unicode	3
1.2 Создание проекта	3
1.2.1 Создайте страницу «Hello, World»	3
1.2.2 Создание репозитория и добавление файла в репозиторий	3
1.2.3 Проверка состояние репозитория	4
1.3 Внесение изменений	4
1.3.1 Измените страницу «Hello, World»	4
1.4 Индексация изменений	6
1.4.1 Коммит изменений	6
1.4.2 Добавьте стандартные теги страницы	7
1.4.3 История	10
1.4.4 Получение старых версий	11
1.4.5 Создание тегов версий	12
1.8 Удаление коммитов из ветки	13
1.9 Работа с ветками	14
1.9.1 Создайте ветку	14
1.9.2 Просмотрите текущие ветки	14
1.9.3 Слияние веток	16
1.9.4 Сброс ветки style	16
1.9.5 Перебазирование	17
Вывод	18

Цель работы

Целью данной работы является изучение работы с git, а именно знакомство с командами, с помощью которых мы сможем работать с репозиториями, файлами, папками и ветками.

Теоретическое введение

Git (произносится «гит») — распределённая система управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года; координатор — Дзюн Хамано.

Среди проектов, использующих Git, — ядро Linux, Swift, Android, Drupal, Cairo, GNU Core Utilities, Mesa, Wine, Chromium, Compiz Fusion, FlightGear, jQuery, PHP, NASM, MediaWiki, DokuWiki, Qt, ряд дистрибутивов Linux.

Программа является свободной и выпущена под лицензией GNU GPL версии 2. По умолчанию используется TCP-порт 9418.

Выполнение лабораторной работы

1.1 Подготовка

1.1.1 Установка имени и электронной почты

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your_email@whatever.com"
```

1.1.2 Параметры установки окончаний строк

Настройка `core.autocrlf` с параметрами `true` и `input` делает все переводы строк текстовых файлов в главном репозитории одинаковыми. `core.autocrlf true` - git автоматически конвертирует CRLF->LF при коммите и обратно LF->CRLF при выгрузке кода из репозитория на файловую систему (используют в Windows). `core.autocrlf input` - конвертация CRLF в LF только при коммитах (используют в Mac/Linux).

Если `core.safecrlf` установлен в `true` или `warn`, git проверяет, если преобразование является обратимым для текущей настройки `core.autocrlf`. `core.safecrlf true` - отвержение необратимого преобразования lf<->crlf.

Полезно, когда специфические бинарники похожие на текстовые файлы. `core.safecrlf warn` - печать только предупреждение, но принимает необратимый переход.

Для пользователей Windows:

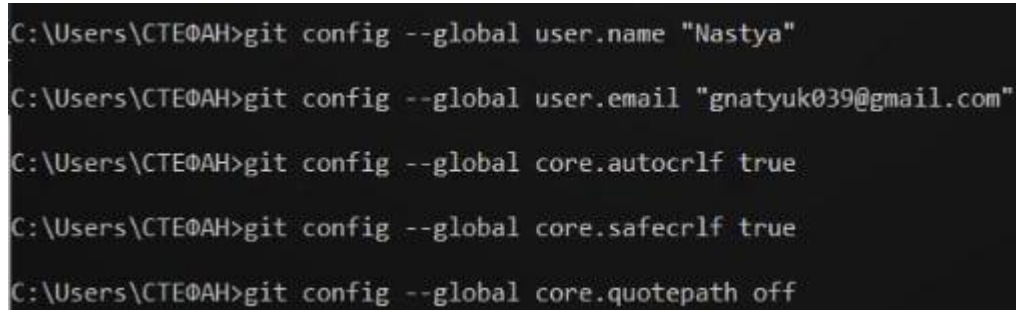
```
git config --global core.autocrlf true
```

```
git config --global core.safecrlf true
```

1.1.3 Установка отображения unicode

По умолчанию, git будет печатать не-ASCII символов в именах файлов в виде восьмеричных последовательностей. Что бы избежать нечитаемых строк, установите соответствующий флаг.

```
git config --global core.quotepath off
```



```
C:\Users\СТЕФАН>git config --global user.name "Nastya"
C:\Users\СТЕФАН>git config --global user.email "gnatyuk039@gmail.com"
C:\Users\СТЕФАН>git config --global core.autocrlf true
C:\Users\СТЕФАН>git config --global core.safecrlf true
C:\Users\СТЕФАН>git config --global core.quotepath off
```

Рис.1:

1.2 Создание проекта

1.2.1 Создайте страницу «Hello, World»

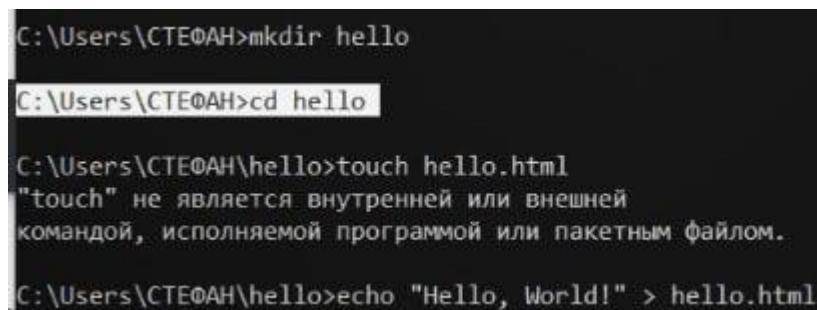
Начните работу в пустом рабочем каталоге с создания пустого каталога с именем hello, затем войдите в него и создайте там файл с именем hello.html.

```
mkdir hello
```

```
cd hello
```

```
touch hello.html
```

```
echo "Hello, World!" > hello.html
```



```
C:\Users\СТЕФАН>mkdir hello
C:\Users\СТЕФАН>cd hello
C:\Users\СТЕФАН\hello>touch hello.html
"touch" не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
C:\Users\СТЕФАН\hello>echo "Hello, World!" > hello.html
```

Рис.2:

1.2.2 Создание репозитория и добавление файла в репозиторий

Добавим файл в репозиторий.

```
git add hello.html
```

```
git commit -m "Initial Commit"
```

```
C:\Users\CTEΦAH\hello>git init
Initialized empty Git repository in C:/Users/CTEΦAH/hello/.git/

C:\Users\CTEΦAH\hello>git add hello.html

C:\Users\CTEΦAH\hello>git commit -m "Initial Commit"
[master (root-commit) 957d9f5] Initial Commit
1 file changed, 1 insertion(+)
create mode 100644 hello.html
```

Рис.3:

1.2.3 Проверка состояния репозитория

Используйте команду `git status`, чтобы проверить текущее состояние репозитория.

`git status`

```
C:\Users\CTEΦAH\hello>git commit -m "Initial Commit"
[master (root-commit) 957d9f5] Initial Commit
1 file changed, 1 insertion(+)
create mode 100644 hello.html

C:\Users\CTEΦAH\hello>git status
On branch master
nothing to commit, working tree clean

C:\Users\CTEΦAH\hello>git status
On branch master
```

Рис.4:

1.3 Внесение изменений

1.3.1 Измените страницу «Hello, World»

Добавим кое-какие HTML-теги к нашему приветствию. Измените содержимое файла `hello.html` на:

```
<h1>Hello, World!</h1>
```

Рис.6:

Проверьте состояние рабочего каталога.

`git status`

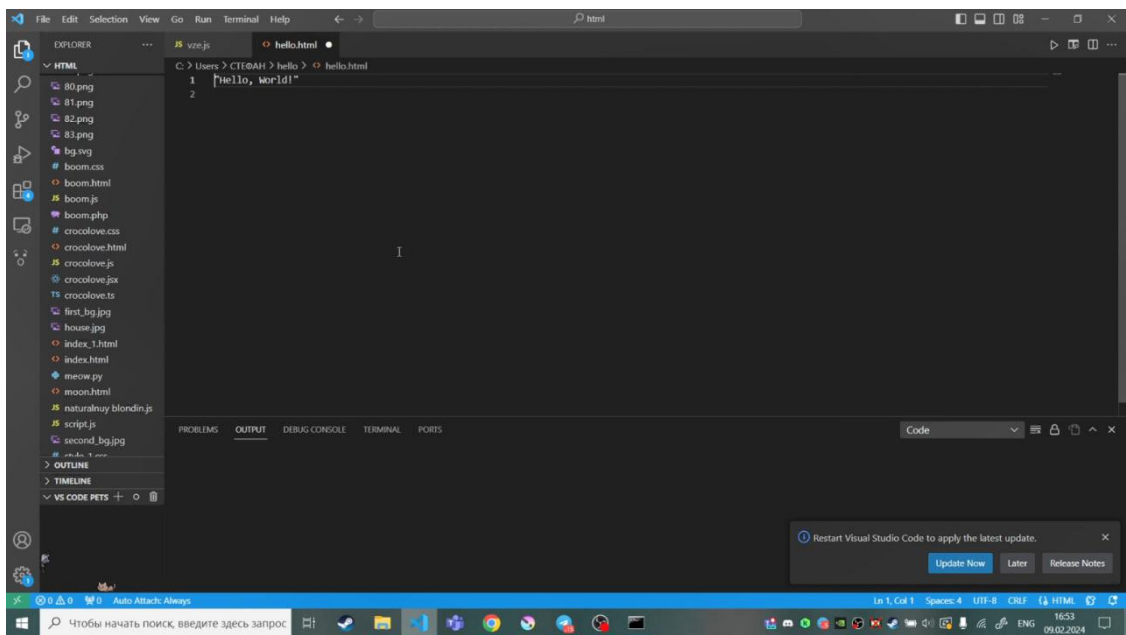


Рис.5:

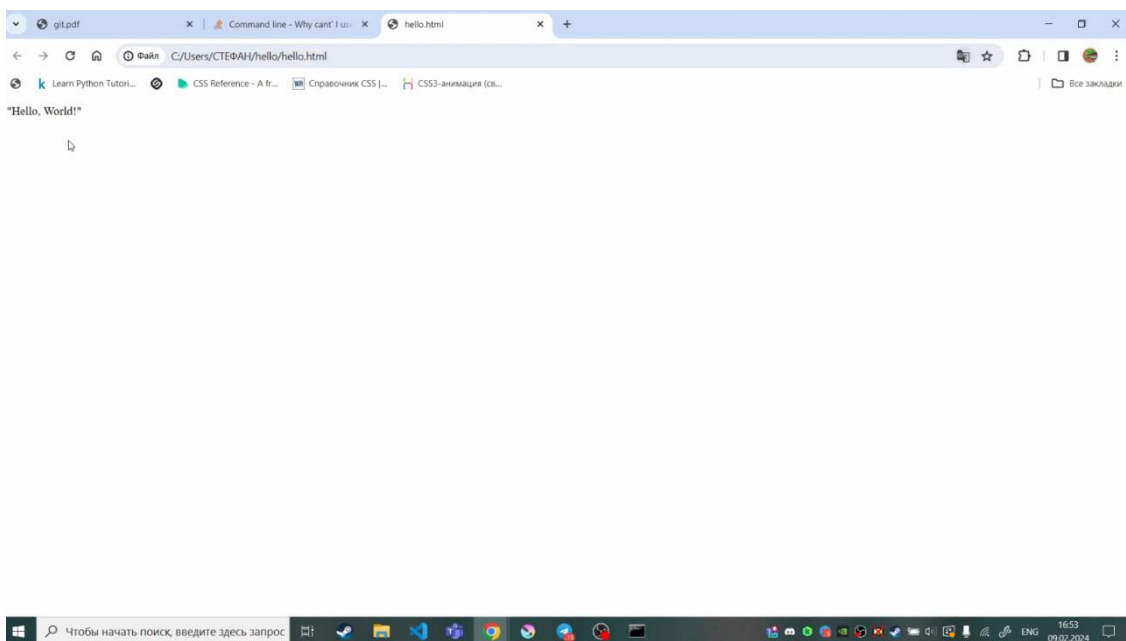


Рис.7:

```
C:\Users\СТЕФАН\hello>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\СТЕФАН\hello>git checkout hello.html
Updated 1 path from the index

C:\Users\СТЕФАН\hello>git status
On branch master
nothing to commit, working tree clean
```

Рис.8:

1.4 Индексация изменений

1.4.1 Коммит изменений

Теперь выполните команду git, чтобы проиндексировать изменения. Проверьте состояние.

git add hello.html

git status

```
C:\Users\СТЕФАН\hello>git add hello.html

C:\Users\СТЕФАН\hello>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

C:\Users\СТЕФАН\hello>
```

Рис.9:

Сделайте коммит и проверьте состояние.

git commit

Откроется редактор.

В первой строке введите комментарий: «Added h1 tag». Сохраните файл и выйдите из редактора (для этого в редакторе по-умолчанию (Vim) вам нужно нажать клавишу ESC, ввести :wq и нажать Enter).

Теперь еще раз проверим состояние.

git status

Рабочий каталог чистый, можно продолжить работу

```
# Added h1 tag
Aborting commit due to empty commit message.
# On branch master
C:\Users\СТЕФАН\hello>git status
On branch master:   hello.html
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html
```

Рис.10:

1.4.2 Добавьте стандартные теги страницы

Измените страницу «Hello, World», чтобы она содержала стандартные теги html и body.

```
<html>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Рис.11:

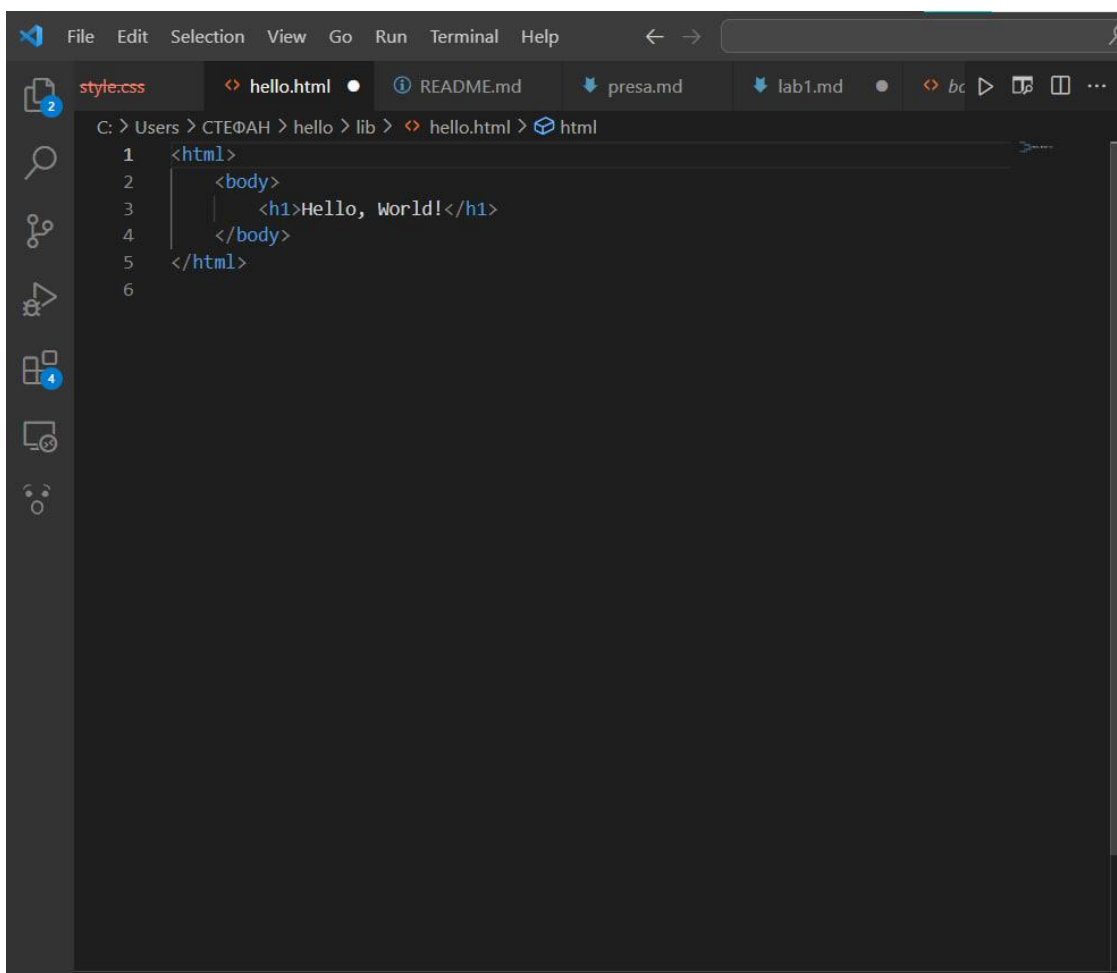


Рис.12:

Теперь добавьте это изменение в индекс git.

```
git add hello.html
```

Теперь добавьте заголовки HTML (секцию head) к странице «Hello, World».

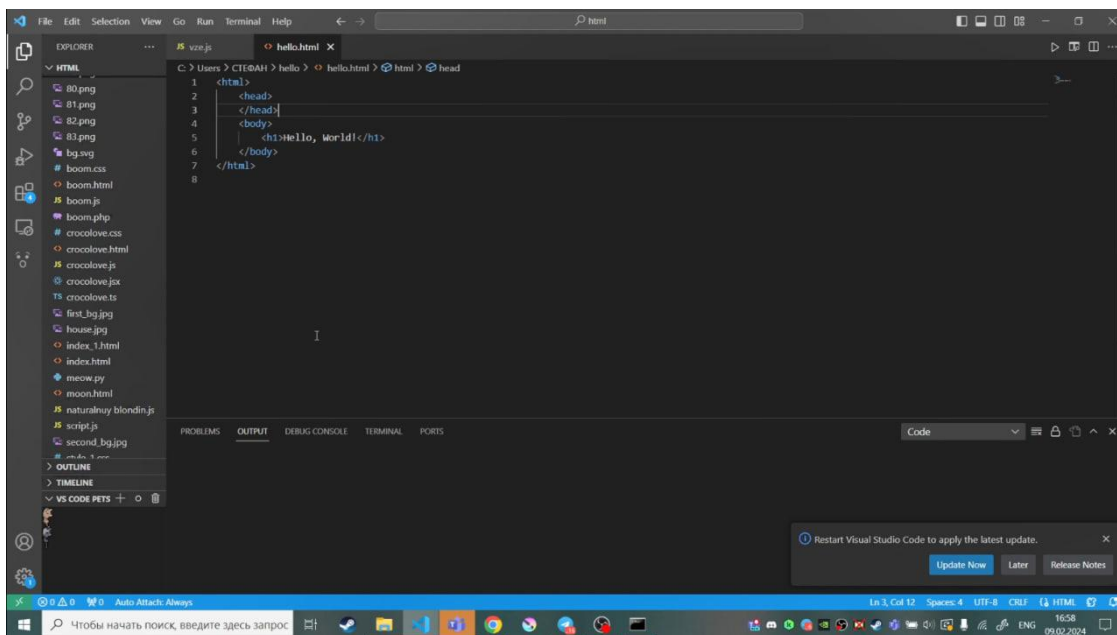


Рис.13:

Проверьте текущий статус:

git status

Обратите внимание на то, что hello.html указан дважды в состоянии. Первое изменение (добавление стандартных тегов) проиндексировано и готово к коммиту. Второе изменение (добавление заголовков HTML) является непроиндексированным. Если бы вы делали коммит сейчас, заголовки не были бы сохранены в репозиторий.

```
C:\Users\СТЕФАН\hello>git add hello.html

C:\Users\СТЕФАН\hello>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html
```

Рис.14:

Произведите коммит проиндексированного изменения (значение по умолчанию), а затем еще раз проверьте состояние.

git commit -m "Added standard HTML page tags"

git status

```

C:\Users\CTEΦAH\hello>git commit -m "Added standard HTML page tags"
[master 5e0e48d] Added standard HTML page tags
1 file changed, 5 insertions(+), 1 deletion(-)

C:\Users\CTEΦAH\hello>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   hello.html

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\CTEΦAH\hello>

```

Рис.15:

Состояние команды говорит о том, что hello.html имеет незафиксированные изменения, но уже не в буферной зоне.

Теперь добавьте второе изменение в индекс, а затем проверьте состояние с помощью команды git status.

git add .

git status

```

C:\Users\CTEΦAH\hello>git add .

C:\Users\CTEΦAH\hello>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   hello.html

C:\Users\CTEΦAH\hello>git commit -m "Added HTML header"
[master 726eb24] Added HTML header
1 file changed, 2 insertions(+)

C:\Users\CTEΦAH\hello>

```

Рис.16:

1.4.3 История

Получим список произведенных изменений:

git log

```

C:\Users\CTE0AH\hello>git log
commit 35a2438ebc6e4b9a3b2b06c9c783a09994de1203 (HEAD -> master)
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 17:19:44 2024 +0300

    Revert "Oops, we didn't want this commit"
    This reverts commit 40de3c54b9e1c464894ae3f3edcecd30f7d3a9a.

commit 40de3c54b9e1c464894ae3f3edcecd30f7d3a9a
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 17:19:00 2024 +0300

    Oops, we didn't want this commit

commit 726eb24c19385f9b560bf7f9c2d589b47c3ca71 (tag: v1)
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 17:00:59 2024 +0300

    Added HTML header

commit 5e0e48d35aa8a9d4880ed3aadb45f7e2c7b70c7 (tag: v1-beta)
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 16:59:35 2024 +0300

    Added standard HTML page tags

commit 937d9f5f3d35317302353d542c71b3465aec1
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 16:43:28 2024 +0300

    Initial Commit
  
```

Рис.17:

1.4.4 Получение старых версий

Получите хэши предыдущих версий

git log

Изучите данные лога и найдите хэш для первого коммита. Он должен быть в последней строке данных. Используйте этот хэш-код (достаточно первых 7 знаков) в команде ниже. Затем проверьте содержимое файла hello.html.

git checkout "hash"

cat hello.html

```

C:\Users\CTE0AH\hello>git checkout 726eb24
Note: switching to '726eb24'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false

HEAD is now at 726eb24 Added HTML header
C:\Users\CTE0AH\hello>
  
```

Рис.18:

Вернитесь к последней версии в ветке master

git checkout master

cat hello.html

```

C:\Users\CTE0AH\hello>git checkout master
Switched to branch 'master'
  
```

Рис.19:

master — имя ветки по умолчанию. Переключая имена веток, вы попадаете на последнюю версию выбранной ветки.

1.4.5 Создание тегов версий

Давайте назовем текущую версию страницы hello первой (v1).

Создайте тег первой версии

```
git tag v1
```

Теперь текущая версия страницы называется v1.

Теги для предыдущих версий Давайте создадим тег для версии, которая идет перед текущей версией и назовем его v1-beta. В первую очередь нам надо переключиться на предыдущую версию. Вместо поиска до хэш, мы будем использовать ^, обозначающее «родитель v1». Вместо обозначения v1^ можно использовать v1~1. Это обозначение можно определить как «первую версию предшествующую v1».

```
git checkout v1^
```

```
cat hello.html
```

Это версия с тегами html и body, но еще пока без head. Давайте сделаем ее версией v1-beta.

```
git tag v1-beta
```

```
C:\Users\СТЕФАН\hello>git tag v1-beta
C:\Users\СТЕФАН\hello>git checkout v1
Previous HEAD position was 5e0e48d Added standard HTML page tags
HEAD is now at 726eb24 Added HTML header
C:\Users\СТЕФАН\hello>git checkout v1-beta
Previous HEAD position was 726eb24 Added HTML header
HEAD is now at 5e0e48d Added standard HTML page tags
C:\Users\СТЕФАН\hello>git checkout v1^~1
Note: switching to 'v1~1'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

    git switch -c <new-branch-name>

Or undo this operation with:

    git switch -

Turn off this advice by setting config variable advice.detachedHead to false
HEAD is now at 5e0e48d Added standard HTML page tags
C:\Users\СТЕФАН\hello>git tag v1-beta
```

1.8 Удаление коммитов из ветки

`git revert` является мощной командой, которая позволяет отменить любые коммиты в репозиторий. Однако, и оригинальный и «отмененный» коммиты видны в истории ветки (при использовании команды `git log`).

Часто мы делаем коммит, и сразу понимаем, что это была ошибка. Было бы неплохо иметь команду «возврата», которая позволила бы нам сделать вид, что неправильного коммита никогда и не было. Команда «возврата» даже предотвратила бы появление нежелательного коммита в истории `git log`.

Давайте сделаем быструю проверку нашей истории коммитов. Выполните:

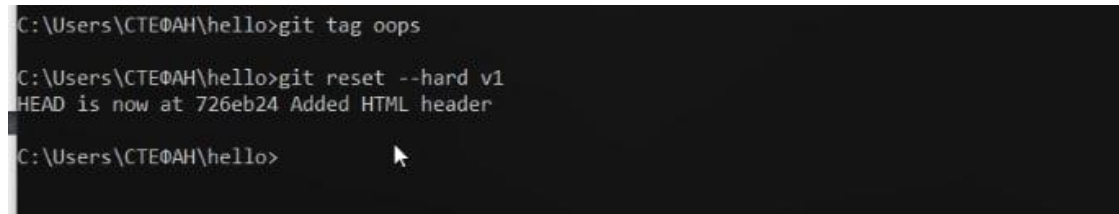
```
git log
```

Мы видим, что два последних коммита в этой ветке — «Oops» и «Revert Oops».

Давайте удалим их с помощью сброса.

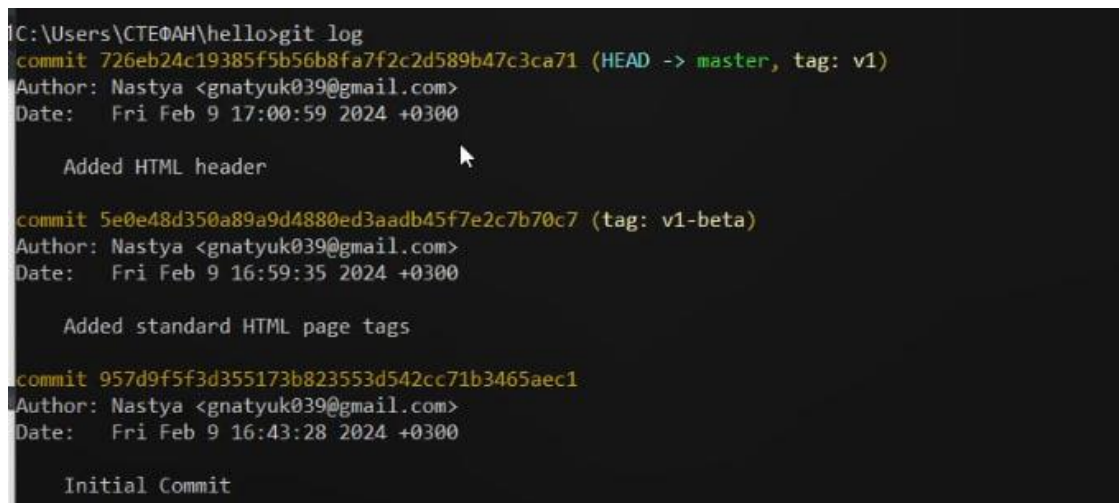
Но прежде чем удалить коммиты, давайте отметим последний коммит тегом, чтобы потом можно было его найти.

```
git tag oops
```



```
C:\Users\СТЕФАН\hello>git tag oops
C:\Users\СТЕФАН\hello>git reset --hard v1
HEAD is now at 726eb24 Added HTML header
C:\Users\СТЕФАН\hello>
```

Рис.22:



```
C:\Users\СТЕФАН\hello>git log
commit 726eb24c19385f5b56b8fa7f2c2d589b47c3ca71 (HEAD -> master, tag: v1)
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 17:00:59 2024 +0300

    Added HTML header

commit 5e0e48d350a89a9d4880ed3aadb45f7e2c7b70c7 (tag: v1-beta)
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 16:59:35 2024 +0300

    Added standard HTML page tags

commit 957d9f5f3d355173b823553d542cc71b3465aec1
Author: Nastya <gnatyuk039@gmail.com>
Date:   Fri Feb 9 16:43:28 2024 +0300

    Initial Commit
```

Рис.23:

Глядя на историю лога, мы видим, что коммит с тегом «v1» является коммитом, предшествующим ошибочному коммиту. Давайте сбросим ветку до этой точки.

Поскольку ветка имеет тег, мы можем использовать имя тега в команде сброса (если она не имеет тега, мы можем использовать хэш-значение).

```
git reset --hard v1
```

```
git log
```

Наша ветка master теперь указывает на коммит v1, а коммитов Oops и Revert Oops в ветке уже нет. Параметр --hard указывает, что рабочий каталог должен быть обновлен в соответствии с новым head ветки.

1.9 Работа с ветками

Пора сделать наш hello world более выразительным. Так как это может занять некоторое время, лучше переместить эти изменения в отдельную ветку, чтобы изолировать их от изменений в ветке master.

1.9.1 Создайте ветку

Давайте назовем нашу новую ветку «style».

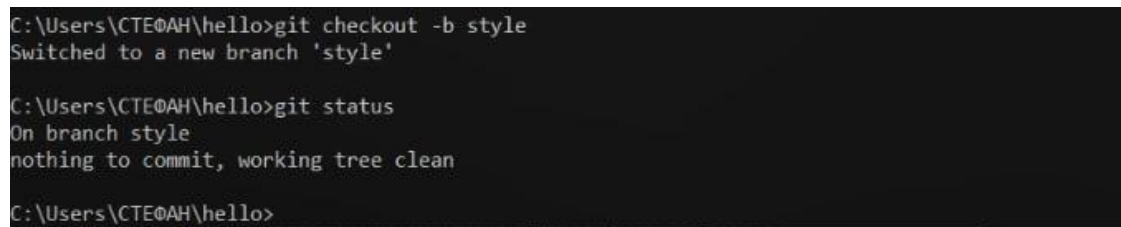
Выполните:

```
git checkout -b style
```

```
git status
```

```
git checkout -b
```

Обратите внимание, что команда git status сообщает о том, что вы находитесь в ветке «style».



```
C:\Users\CTE\AN\hello>git checkout -b style
Switched to a new branch 'style'

C:\Users\CTE\AN\hello>git status
On branch style
nothing to commit, working tree clean

C:\Users\CTE\AN\hello>
```

Рис.24:

1.9.2 Просмотрите текущие ветки

Теперь у нас в репозитории есть две отличающиеся ветки. Используйте следующую лог-команду для просмотра веток и их отличий.

Выполните:

```
git log --graph --all
```



```

C:\Users\CTE\AH\hello>git log --graph --all
*   commit f2345f4a33a2b135f4f85ee5a319e3f9dc07d522 (HEAD -> style)
| \ Merge: 2e3ed29 6ae920c
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 18:05:25 2024 +0300
|
|      Merge branch 'master' into style
|
*   commit 6ae920c36556ee23be1d81952fb0373180016a58 (master)
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 18:02:32 2024 +0300
|
|      Added README
|
*   commit 2e3ed29f672c4e45189a6140cff9462fdf77151a
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:58:32 2024 +0300
|
|      Updated index.html
|
*   commit e1ef7116e3e80205575d82fc1bf13dd965d5d852
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:57:30 2024 +0300
|
|      Hello uses style.css
|
*   commit 0f99f12a8eac0bd6f908515ce4c02c51d3fc4cc9
| / Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:55:47 2024 +0300
|
|      Added css stylesheet
|
*   commit a256234265ec8f6d8cb4a6601f33bb438be37dc9
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:39:35 2024 +0300
|
|      Added index.html.
|
*   commit 77659900c2ef81a42604845e0014faf4eebc4ea4
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:35:59 2024 +0300
|
|      Moved hello.html to lib
|
*   commit e862f7e23d260f15f965bfa027646eefa0aa2e02
|  Author: Nastya <gnatyuk039@gmail.com>
|  Date:   Fri Feb 9 17:30:53 2024 +0300
|
|      Add an author/email comment

```

Puc.25:

1.9.3 Слияние веток

Слияние переносит изменения из двух веток в одну. Давайте вернемся к ветке style и сольем master с style.

Выполните:

git checkout style

git merge master

git log --graph --all

```
C:\Users\СТЕФАН\hello>git checkout style
Switched to branch 'style'

C:\Users\СТЕФАН\hello>git merge master
Merge made by the 'ort' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

C:\Users\СТЕФАН\hello>cd he_

C:\Users\СТЕФАН\hello>git log --graph --all
*   commit f2345f4a33a2b135f4f85ee5a319e3f9dc07d522 (style)
   | Merge: 2e3ed29 6ae920c
   | Author: Nastya <gnatyuk039@gmail.com>
   | Date:   Fri Feb 9 18:05:25 2024 +0300
   |
   | Merge branch 'master' into style
   |
   | *   commit 6ae920c36556ee23be1d81952fb0373180016a58 (HEAD -> master)
   |   | Author: Nastya <gnatyuk039@gmail.com>
   |   | Date:   Fri Feb 9 18:02:32 2024 +0300
   |   |
   |   | Added README
   |   |
   |   | *   commit 2e3ed29f672c4e45189a6140cff9462fdf77151a
   |   |   | Author: Nastya <gnatyuk039@gmail.com>
   |   |   | Date:   Fri Feb 9 17:58:32 2024 +0300
   |   |   |
   |   |   | Updated index.html
   |   |   |
   |   |   | *   commit e1ef7116e3e80205575d82fc1bf13dd965d5d852
   |   |   |   | Author: Nastya <gnatyuk039@gmail.com>
   |   |   |   | Date:   Fri Feb 9 17:57:30 2024 +0300
   |   |   |   |
   |   |   |   | Hello uses style.css
   |   |   |   |
   |   |   |   | *   commit 0f99f12a8eac0bd6f908515ce4c02c51d3fc4cc9
   |   |   |   |   | Author: Nastya <gnatyuk039@gmail.com>
   |   |   |   |   | Date:   Fri Feb 9 17:55:47 2024 +0300
   |   |   |   |   |
   |   |   |   |   |
```

Путем периодического слияния ветки master с веткой style вы можете переносить из master любые изменения и поддерживать совместимость изменений style с изменениями в основной ветке.

1.9.4 Сброс ветки style

Вернемся на ветке style к точке перед тем, как мы слили ее с веткой master. Мы можем сбросить ветку к любому коммиту. По сути, это изменение указателя ветки на любую точку дерева коммитов.

В этом случае мы хотим вернуться в ветке style в точку перед слиянием с master.

Нам необходимо найти последний коммит перед слиянием.

Выполните:

```
git checkout style
```

```
git log --graph
```

Мы видим, что коммит «Updated index.html» был последним на ветке style перед слиянием. Давайте сбросим ветку style к этому коммиту. Выполните:

```
git reset --hard "hash"
```

```
C:\Users\СТЕФАН\hello>git reset --hard 2e3ed29
HEAD is now at 2e3ed29 Updated index.html

C:\Users\СТЕФАН\hello>git log --graph --all
* commit 144fa9862400caefd0d3d98f20623ba4fba5bfbe (master)
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 18:13:04 2024 +0300

    Merged master fixed conflict.

* commit 6ae920c36556ee23be1d81952fb0373180016a58
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 18:02:32 2024 +0300

    Added README

* commit 2e3ed29f672c4e45189a6140cff9462fdf77151a (HEAD -> style)
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:58:32 2024 +0300

    Updated index.html

* commit e1ef7116e3e80205575d82fc1bf13dd965d5d852
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:57:30 2024 +0300

    Hello uses style.css

* commit 0f99f12a8eac0bd6f908515ce4c02c51d3fc4cc9
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:55:47 2024 +0300

    Added css stylesheet
```

Рис.28:

1.9.5 Перебазирование

Используем команду rebase вместо команды merge. Мы вернулись в точку до первого слияния и хотим перенести изменения из ветки master в нашу ветку style.

На этот раз для переноса изменений из ветки master мы будем использовать команду git rebase вместо слияния.

Выполните:

git checkout style

git rebase master

git log --graph

```
C:\Users\CTE0AH\hello>git checkout style
Switched to branch 'style'

C:\Users\CTE0AH\hello>git rebase master
Successfully rebased and updated refs/heads/style.

C:\Users\CTE0AH\hello>git log --graph
* commit 1df1f993835ff87c7e62a7919b4044b2c7f5c74c (HEAD -> style)
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:58:32 2024 +0300

    Updated index.html

* commit 2395f2d60a4aafbe2b81af58400719eedd0bb4c6
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:57:30 2024 +0300

    Hello uses style.css

* commit 371a9f9a56d5d1ced584cff4dd02c0c196149b96
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:55:47 2024 +0300

    Added css stylesheet

* commit 6ae920c36556ee23be1d81952fb0373180016a58 (master)
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 18:02:32 2024 +0300

    Added README

* commit a256234265ec8f6d8cb4a6601f33bb438be37dc9
  Author: Nastya <gnatyuk039@gmail.com>
  Date:   Fri Feb 9 17:39:35 2024 +0300

    Added index.html.
```

Рис.29:

Вывод

Мы поработали с git, изучили и попробовали некоторые команды.