ЛАБОРАТОРНАЯ РАБОТА №4

Создание ветки

Пришло время сделать нашу страницу более стильной с помощью CSS. Мы будем развивать эту возможность в новой ветке под названием style.

Выполните

```
git switch -c style
git status
```

Старожилы могут возразить, что их учили создавать ветки командой git checkout -b style. Помните, я упоминал, что команда checkout перегружена функциями и флагами? Старый способ все еще работает, но он не рекомендуется. Новая команда git switch более выразительна и менее восприимчива к ошибкам. Кроме того, в ней меньше флагов и опций, поэтому ее легче запомнить.

Результат

```
$ git switch -c style
Switched to a new branch 'style'
$ git status
On branch style
nothing to commit, working tree clean
```

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

Добавьте файл стилей style.css

Выполните

```
touch style.css
```

Выполните

```
git add style.css
git commit -m "Added css stylesheet"
```

Измените hello.html**, чтобы он использовал** style.css

Файл: hello.html

```
<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Выполните

```
git add hello.html
git commit -m "Included stylesheet into hello.html"
```

Теперь у нас есть новая ветка под названием style с двумя новыми коммитами. Далее мы узнаем, как переключаться между ветками.

Переключение веток

Теперь в вашем проекте есть две ветки:

Выполните

```
git log --all
```

Результат

```
$ git log --all
9288a33 2023-11-28 | Added copyright statement with email (main) [Alexander Shvets]
903eb1d 2023-11-28 | Included stylesheet into hello.html (HEAD -> style) [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Переключение на ветку main

Просто используйте команду git switch для переключения между ветками.

Выполните

```
git switch main
cat hello.html
```

Результат

```
$ git switch main
Switched to branch 'main'
$ cat hello.html
<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Теперь мы находимся в ветке main. Как видите, в hello.html нет никаких следов style.css. Не волнуйтесь, эти изменения все еще есть в репозитории, но мы не можем увидеть их из ветки main.

Вернемся к ветке style

Выполните

```
git switch style
cat hello.html
```

Результат

```
$ git switch style
Switched to branch 'style'
$ cat hello.html
<!-- Author: Alexander Shvets (alex@githowto.com) -->
<html>
  <head>
    <link type="text/css" rel="stylesheet" media="all" href="style.css" />
  </head>
  <body>
    <h1>Hello, World!</h1>
  </body>
</html>
```

Мы вернулись к ветке style. Как видите, наши изменения, связанные с CSS, присутствуют.

Перемещение файлов

Мы давольны нашими CSS-изменениями, но есть только один момент, который я хотел бы решить до того, как мы объединим наши изменения с main. Давайте переименуем файл hello.html в index.html. Также давайте перенесем наш файл стилей в специально отведенную директорию css.

Просмотр истории изменений в конкретном файле

Git позволяет просматривать историю изменений конкретного файла. Давайте посмотрим историю изменений файла hello.html перед его переименованием.

Выполните

```
git log hello.html
git log style.css
```

Результат

```
$ git log hello.html
903eb1d 2023-11-28 | Included stylesheet into hello.html (HEAD -> style) [Alexander Shvets]
9288a33 2023-11-28 | Added copyright statement with email (main) [Alexander Shvets]
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
5836970 2023-11-28 | Initial commit [Alexander Shvets]
$ git log style.css
555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
```

Просмотр различий для конкретного файла

Возможность просмотра истории изменений для конкретного файла очень полезна. Она позволяет увидеть, что именно изменилось, а также кто и когда внес эти изменения. Кроме того, существует возможность увидеть изменения, связанные с конкретным коммитом. Я постоянно пользуюсь этим, чтобы разобраться, почему та или иная штука была реализована именно так в настоящей версии кода.

Команда show используется для просмотра изменений в конкретном коммите. Посмотрим изменения в файле hello.html в коммите, с тегом v1 (можно использовать любую ссылку на коммит, например, метку HEAD, хеш коммита, имя ветки или тега и т.д.).

Выполните

```
git show v1
```

Результат

```
$ git show v1
b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]

diff --git a/hello.html b/hello.html
index 6da0629..0d576c4 100644
--- a/hello.html
+++ b/hello.html
@@ -1,4 +1,6 @@
 <html>
+  <head>
+  </head>
   <body>
     <h1>Hello, World!</h1>
   </body>
```

Переименуйте файл hello.html

Как видите, очень удобно иметь возможность видеть историю изменений конкретного файла. Но когда вы переименовываете или перемещаете какой-либо файл, есть риск потерять историю этого файла, если вы выполните эту процедуру неправильно.

Давайте переименуем наш файл hello.html в index.html с помощью стандартной команды mv и посмотрим, что из этого получится.

Выполните

```
mv hello.html index.html
git status
```

Результат

```
$ mv hello.html index.html
$ git status
On branch style
Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    deleted:    hello.html

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    index.html

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

Git воспринимает наше изменение так, будто файл был удален и создан заново. Это тревожный звоночек. Нам нужно сообщить Git, что мы именно переименовали файл, а не удалили его и сразу создали новый. В простейшем случае Git сам поймёт, что файл был переименован, как только мы добавим его в индекс:

Выполните

```
git add .
git status
```

Результат

```
$ git add .
$ git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    hello.html -> index.html
```

Видите, файл указан как переименованный. Но это всего лишь Git пытается проявить смекалку. Это не всегда работает. Например, если вы переименовали, **а также** изменили кучу файлов, Git может оказаться не в состоянии понять, что именно было переименовано. В этом случае вы можете потерять информацию об истории файлов до их переименования, поскольку файлы будут восприняты как новые.

Переместите style.css **безопасным способом**

В большинстве операционных систем переименование и перемещение файлов — это одно и то же. Итак, давайте переместим наш файл style.css в директорию css, но на этот раз сделаем это безопасно с помощью команды git mv. Эта команда гарантирует, что перемещение будет записано в истории Git как перемещение.

Выполните

```
mkdir css
git mv style.css css/style.css
git status
```

Результат

```
$ mkdir css
$ git mv style.css css/style.css
$ git status
On branch style
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    renamed:    style.css -> css/style.css
    renamed:    hello.html -> index.html
```

Давайте закоммитим наши изменения и проверим историю изменений в файле css/styles.css. Для просмотра истории файла до его перемещения нам потребуется добавить опцию --follow. Выполним оба варианта команды, чтобы понять разницу.

Выполните

```
git commit -m "Renamed hello.html; moved style.css"
git log css/style.css
git log --follow css/style.css
```

Результат

```
$ git commit -m "Renamed hello.html; moved style.css"
[style 0ee0113] Renamed hello.html; moved style.css
 2 files changed, 0 insertions(+), 0 deletions(-)
 rename style.css => css/style.css (100%)
 rename hello.html => index.html (100%)
$ git log css/style.css
0ee0113 2023-11-28 | Renamed hello.html; moved style.css (HEAD -> style) [Alexander Shvets]
$ git log --follow css/style.css
0ee0113 2023-11-28 | Renamed hello.html; moved style.css (HEAD -> style) [Alexander Shvets]
555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
```

Изменения в ветке main

Как я уже говорил, Git позволяет работать с несколькими ветками одновременно. Это очень удобно при работе в команде, поскольку люди могут параллельно работать над разными функциями. Это также полезно при работе в одиночку: разрабатывая функции в отдельных ветках, вы можете исправлять ошибки и выпускать небольшие обновления, используя стабильный код в ветке main.

Создайте файл README

Создадим файл README. В нем будет рассказано о сути нашего проекта.

Файл: README

```
This is the Hello World example from the GitHowTo tutorial.
```

Сделайте коммит файла README **в ветку** main

В настоящее время мы находимся в ветке style. Файл README не является частью этой ветки, поэтому перед коммитом мы должны переключиться на ветку main.

Выполните

```
git switch main
git add README
git commit -m "Added README"
```

Результат

```
$ git switch main
Switched to branch 'main'
$ git add README
$ git commit -m "Added README"
[main ee16740] Added README
 1 file changed, 1 insertion(+)
 create mode 100644 README
```

ПРОСМОТР ОТЛИЧАЮЩИХСЯ ВЕТОК

Просмотрите текущие ветки

Теперь у нас есть две расходящиеся ветки в репозитории. Используйте следующую команду log для просмотра веток и их расхождения.

Выполните

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

Результат

```
$ git log --all --graph
* ee16740 2023-11-28 | Added README (HEAD -> main) [Alexander Shvets]
| * 0ee0113 2023-11-28 | Renamed hello.html; moved style.css (style) [Alexander Shvets]
| * 903eb1d 2023-11-28 | Included stylesheet into hello.html [Alexander Shvets]
| * 555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
|/
* 9288a33 2023-11-28 | Added copyright statement with email [Alexander Shvets]
* b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
* 46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
* 78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
* 5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

Опция --all гарантирует, что мы видим все ветки, так как по умолчанию в логе показывается только текущая ветка.

Опция --graph добавляет простое дерево коммитов, представленное в виде простых текстовых линий. Мы видим обе ветки (style и main) причём ветка main помечена как HEAD, что означает, что она является текущей. Общим предком для обеих веток является ветка, в которую был внесен коммит «Added copyright statement with email».

Слияние

Слияние веток

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

Выполните

```
git switch style
git merge main
git log --all --graph
```

Результат

```
$ git switch style
Switched to branch 'style'
$ git merge main
Merge made by the 'ort' strategy.
 README | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README
$ git log --all --graph
*   a33deed 2023-11-28 | Merge branch 'main' into style (HEAD -> style) [Alexander Shvets]
|\
| * ee16740 2023-11-28 | Added README (main) [Alexander Shvets]
* | 0ee0113 2023-11-28 | Renamed hello.html; moved style.css [Alexander Shvets]
* | 903eb1d 2023-11-28 | Included stylesheet into hello.html [Alexander Shvets]
* | 555372e 2023-11-28 | Added css stylesheet [Alexander Shvets]
|/
* 9288a33 2023-11-28 | Added copyright statement with email [Alexander Shvets]
* b7614c1 2023-11-28 | Added HTML header (tag: v1) [Alexander Shvets]
* 46afaff 2023-11-28 | Added standard HTML page tags (tag: v1-beta) [Alexander Shvets]
* 78433de 2023-11-28 | Added h1 tag [Alexander Shvets]
* 5836970 2023-11-28 | Initial commit [Alexander Shvets]
```

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

Однако, это делает графики коммитов действительно уродливыми. Позже мы рассмотрим возможность перебазирования, как альтернативы слиянию.

Далее

Но что если изменения в ветке main конфликтуют с изменениями в style?