



HTML и CSS. Адаптивная вёрстка и автоматизация

Уровень 2, с 21 ноября 2022 по 30 января 2023

Меню курса

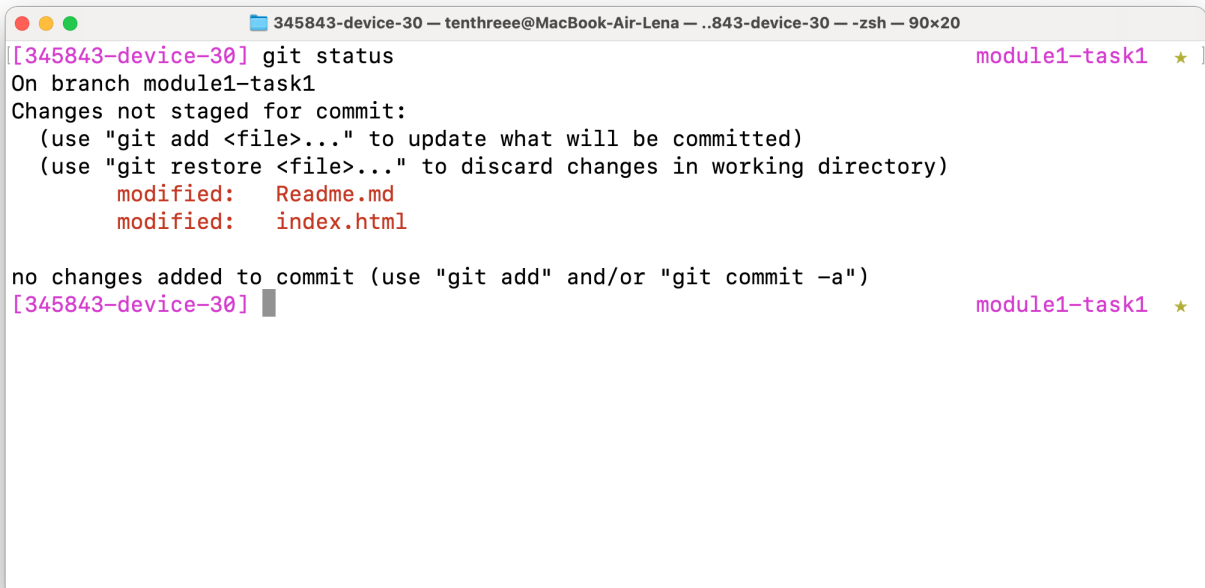
[Главная](#) / [2. Методологии вёрстки](#) /

📖 2.26. Запись изменений в репозиторий

🕒 ~ 6 минут

После того как вы создали ветку и поработали в ней у себя локально, нужно сохранить результат, чтобы он не пропал и в итоге оказался в репозитории.

Для начала введём команду `git status`. Она сообщает, в каком состоянии находится репозиторий: показывает текущее состояние в рабочей ветке, а именно список с названиями изменённых файлов, если они есть, и указывает на те, которые ожидают записи и сохранения (обычно они выделены красным цветом).



```
[345843-device-30] git status                                     module1-task1 ★
On branch module1-task1
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:   Readme.md
        modified:   index.html

no changes added to commit (use "git add" and/or "git commit -a")
[345843-device-30]                                             module1-task1 ★
```

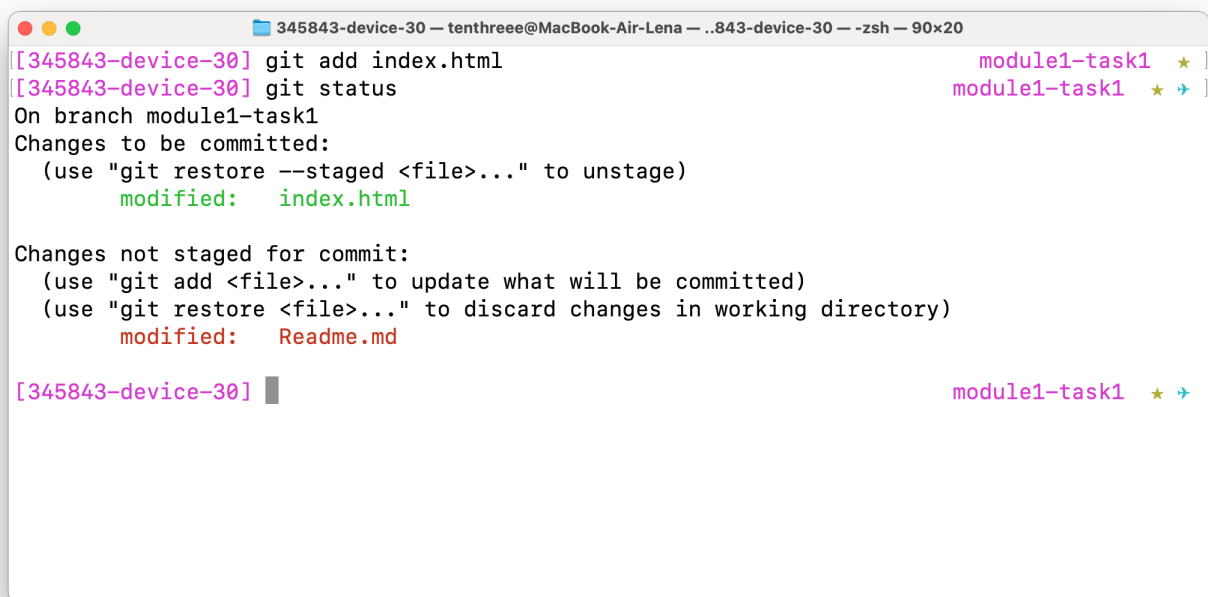
Состояние ветки

— `On branch module1-task1` означает, что мы находимся в ветке `module1-task1`

- `Changes not staged for commit` — это неотслеживаемые изменения (те самые файлы, которые ожидают записи и сохранения).

Чтобы сохранить изменения, нужно их проиндексировать — сообщить Git, что именно мы собираемся сохранять. На скриншоте выше видно, что Git подсказывает, что нужно сделать: `use "git add <file>..." to update what will be committed`.

Для примера сделаем `git add` одного файла и посмотрим, что произошло. Пишем `git add имя-файла`. Если имя файла очень длинное, вы можете начать его писать, затем нажать Tab и терминал сам предложит вам продолжение пути к файлу.



```
[345843-device-30] git add index.html
[345843-device-30] git status
On branch module1-task1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   index.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:   README.md

[345843-device-30]
```

Индексируем index.html

Видим, что файл `index.html` попал в изменения, которые будут закоммичены. Из красного он стал зелёным — это означает, что файл проиндексирован.

Если в будущем вы не захотите перечислять каждый файл, в который были внесены изменения, можно отметить их все одной командой `git add -A`.

Теперь проиндексируем все файлы командой `git add .` — точка означает текущую папку, значит эта команда проиндексирует всю папку. И сразу посмотрим `git status`.

```
345843-device-30 — tenthreee@MacBook-Air-Lena — ..843-device-30 — -zsh — 90x20
[345843-device-30] git add .
[345843-device-30] git status
On branch module1-task1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Readme.md
    modified:   index.html

[345843-device-30]
```

Проиндексировали все файлы

Теперь все файлы проиндексированы и мы можем сделать **КОММИТ**, то есть зафиксировать все сохранённые изменения и дать им название. Это делается с помощью команды `git commit -m "ваше сообщение"`. Параметр `-m` нужен, чтобы прикрепить к коммиту сообщение о том, что именно было изменено. Само сообщение нужно поместить в кавычки `" "`. Текст сообщения должен быть лаконичным и в то же время сообщать, что делает коммит. Например, «добавляет имя наставника в Readme», «вводит функцию сортировки изображений», «правит ошибку в поиске городов на карте».

Сделаем коммит и снова проверим `git status`.

```
345843-device-30 — tenthreee@MacBook-Air-Lena — ..843-device-30 — -zsh — 90x20
[345843-device-30] git add .
[345843-device-30] git status
On branch module1-task1
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   Readme.md
    modified:   index.html

[345843-device-30] git commit -m "Начинает работу над проектом"
[module1-task1 b1082bc] Начинает работу над проектом
2 files changed, 13 insertions(+), 13 deletions(-)
rewrite index.html (63%)
[345843-device-30]
```

Проверка текущего статуса

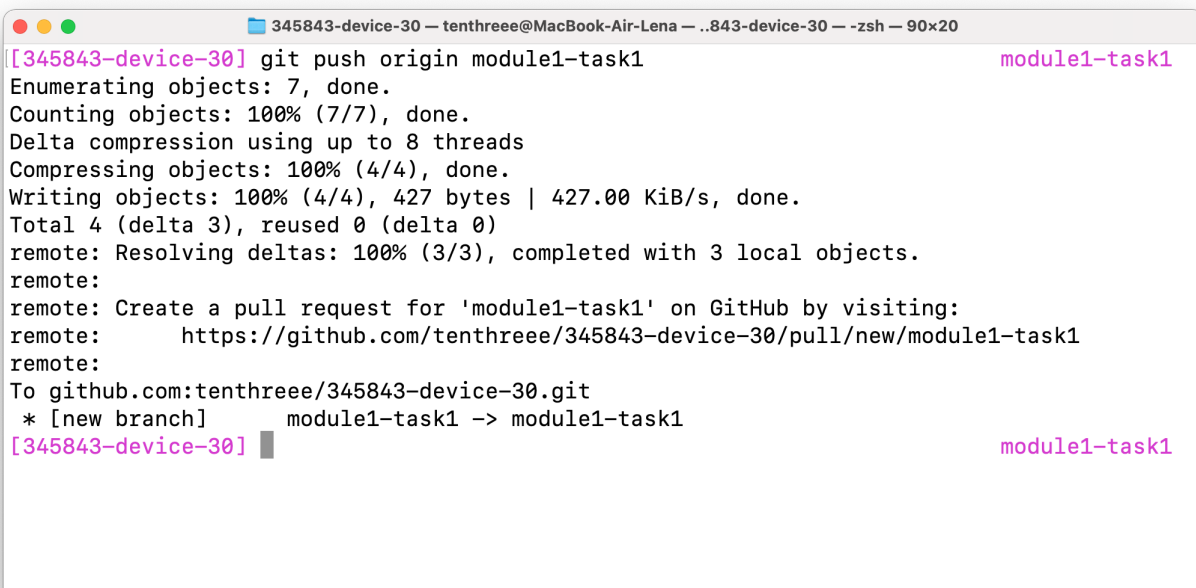
Git пишет, что ему нечего коммитить, потому что текущее состояние уже сохранено. Всё в порядке.

Как часто стоит делать коммиты? Всегда, когда состояние обрело законченный вид. Например, вы добавили новый блок или исправили ошибку, или просто сделали изменения, которые боитесь потерять. Помните: коммит — это просто сохранение.

Итак, сохранения зафиксированы. Значит, они появились в репозитории на GitHub? Пока нет. Дело в том, что в Git коммиты создаются локально, то есть у вас на компьютере. Ветки тоже создаются и сливаются локально. Всё происходит локально. Именно поэтому Git работает очень быстро — не нужно на каждое действие что-то скачивать из интернета или, наоборот, закачивать.

Для отправки локальных изменений (коммитов) в удалённый репозиторий есть команда `push` — толкать. Давайте отправим в репозиторий новый коммит. Введём команду `git push origin module1-task1`, где:

- `origin` — имя удалённого репозитория, который был клонирован на компьютер (ваш форк). Git автоматически называет репозиторий, который вы клонировали, псевдонимом `origin` и помещает в него ссылку, чтобы вам не приходилось каждый раз прописывать путь репозитория.
- `module1-task1` — имя ветки в удалённом репозитории, куда отправить текущую локальную ветку. Если мы не напишем название ветки, то Git по умолчанию отправит все изменения в ветку `master`.



```
345843-device-30 — tenthreee@MacBook-Air-Lena — ..843-device-30 — -zsh — 90x20
[345843-device-30] git push origin module1-task1 module1-task1
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 427 bytes | 427.00 KiB/s, done.
Total 4 (delta 3), reused 0 (delta 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
remote:
remote: Create a pull request for 'module1-task1' on GitHub by visiting:
remote:   https://github.com/tenthreee/345843-device-30/pull/new/module1-task1
remote:
To github.com:tenthreee/345843-device-30.git
 * [new branch]   module1-task1 -> module1-task1
[345843-device-30] module1-task1
```

Отправляем изменения

В этот момент в репозитории на GitHub появится специальная плашка о том, что мы только что отправили изменения и предложение открыть пулреквест.

The screenshot shows the GitHub repository page for 'tentthreee / 345843-device-30', which is a fork of 'htmlacademy-htmlcss/345843-device-30'. The repository has 0 stars, 0 forks, and 1 fork. The main branch is 'master', and there is 1 branch and 0 tags. A yellow banner at the top indicates that 'module1-task1' had recent pushes less than a minute ago, with a green button to 'Compare & pull request'. Below the banner, there are buttons for 'Go to file', 'Add file', and 'Code'. The repository is 1 commit ahead of the upstream 'htmlacademy-htmlcss:master'. A table of files shows the commit history for 'tentthreee' on 25 Jan, with 2 commits. The files listed are 'css', 'fonts', 'img', 'js', '.editorconfig', '.gitattributes', '.gitignore', and 'Contributing.md', all updated 'last month'. On the right side, there are sections for 'About' (Три Десятова), 'Releases' (No releases published), 'Packages' (No packages published), and 'Languages' (HTML 100.0%).

Предложение отправить пулреквест

Чтобы создать пулреквест, нужно нажать зелёную кнопку «Compare & pull request». После этого откроется интерфейс создания пулреквеста, в котором нужно описать, что и зачем мы сделали. Под описанием пулреквеста есть diff — это разница между кодом в мастер-репозитории и кодом, который мы предлагаем.

The screenshot shows the 'Open a pull request' interface on GitHub. At the top, it says 'htmlacademy-htmlcss / 345843-device-30'. Below this, there are tabs for 'Code', 'Pull requests', 'Actions', 'Projects', 'Security', 'Insights', and 'Settings'. The 'Pull requests' tab is selected. The interface shows the 'base repository: htmlacademy-htmlcss/3458...' and 'base: master'. The 'head repository: tentthreee/345843-device-30' and 'compare: module1-task1' are also shown. A green checkmark indicates 'Able to merge. These branches can be automatically merged.' Below this, there is a section for 'Добавляет наставника' (Adds mentor) with a 'Write' and 'Preview' tab. The 'Write' tab is active, showing a text area for the pull request description. To the right of the text area, there are sections for 'Reviewers' (No reviews), 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), and 'Milestone' (No milestone). At the bottom right, there is a green button to 'Create pull request'.

Интерфейс создания пулреквеста

Жмём «Create pull request». Готово, пулреквест открыт.

Если в код потребуется внести изменения, вам нужно снова пройти по цепочке локальные изменения — сохранение — коммит — пуш, только пулреквест заново делать не нужно. Если вы продолжаете вести работу в той же ветке и пулреквест ещё не принят, все ваши изменения автоматически добавятся в пулреквест, созданный из этой ветки, после команды `git push origin название-текущей-ветки`.

Прочитали главу?

Нажмите кнопку «Готово», чтобы сохранить прогресс.

Готово

⚠ Если вы обнаружили ошибку или неработающую ссылку, выделите ее и нажмите Ctrl + Enter

Поиск по материалам

Git

[Все материалы](#)

В самом начале



- ☐ Пройдите опрос
- ☐ Укажите персональные данные
- ☐ Изучите регламент
- ☐ Прочитайте FAQ
- ☐ Добавьте свой Гитхаб
- ☐ Выберите наставника
- ☐ Создайте проект

Мой наставник



[Выбрать наставника](#)

Работа с наставником

У вас осталось **10** из 10 консультаций.

[История](#)



Практикум

Тренажёры

Подписка

Для команд и компаний

Учебник по PHP

Курсы

HTML и CSS. Профессиональная вёрстка сайтов

HTML и CSS. Адаптивная вёрстка и автоматизация

JavaScript. Профессиональная разработка веб-интерфейсов

JavaScript. Архитектура клиентских приложений

React. Разработка сложных клиентских приложений

Node.js. Профессиональная разработка REST API

Node.js и Nest.js. Микросервисная архитектура

TypeScript. Теория типов

Алгоритмы и структуры данных

Паттерны проектирования

Webpack

Vue.js 3. Разработка клиентских приложений

Git и GitHub

Анимация для фронтендеров

Блог

С чего начать

Шпаргалки для разработчиков

Отчеты о курсах

Информация

Об Академии

О центре карьеры

Профессии

Фронтенд-разработчик

JavaScript-разработчик

Фулстек-разработчик

Услуги

Работа наставником

Для учителей

Стать автором

Остальное

Написать нам

Мероприятия

Форум

Соглашение

Конфиденциальность

Сведения об образовательной организации

Лицензия № 4696



Участник

© ООО «Интерактивные обучающие технологии», 2013–2023

