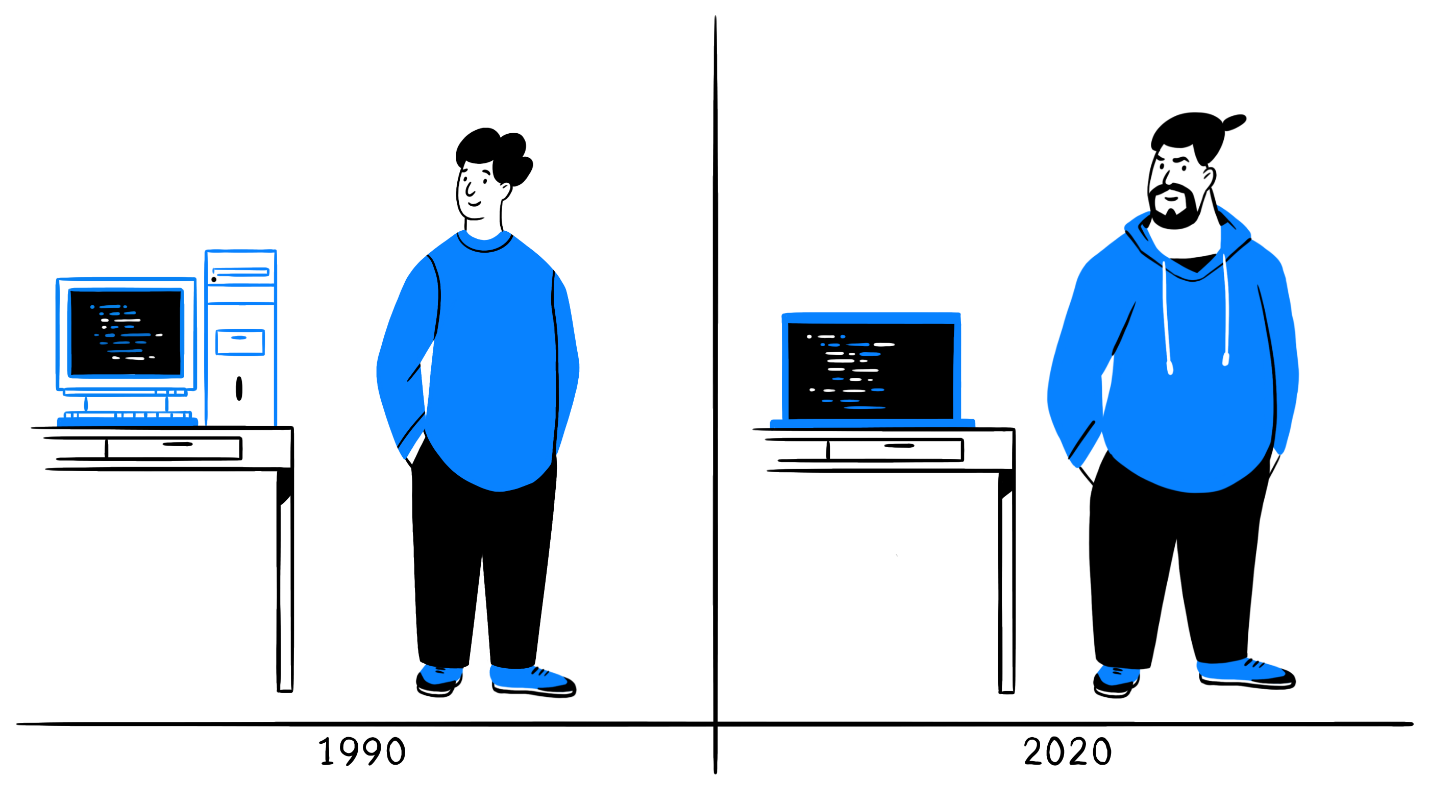
**Bash и Git. Введение**

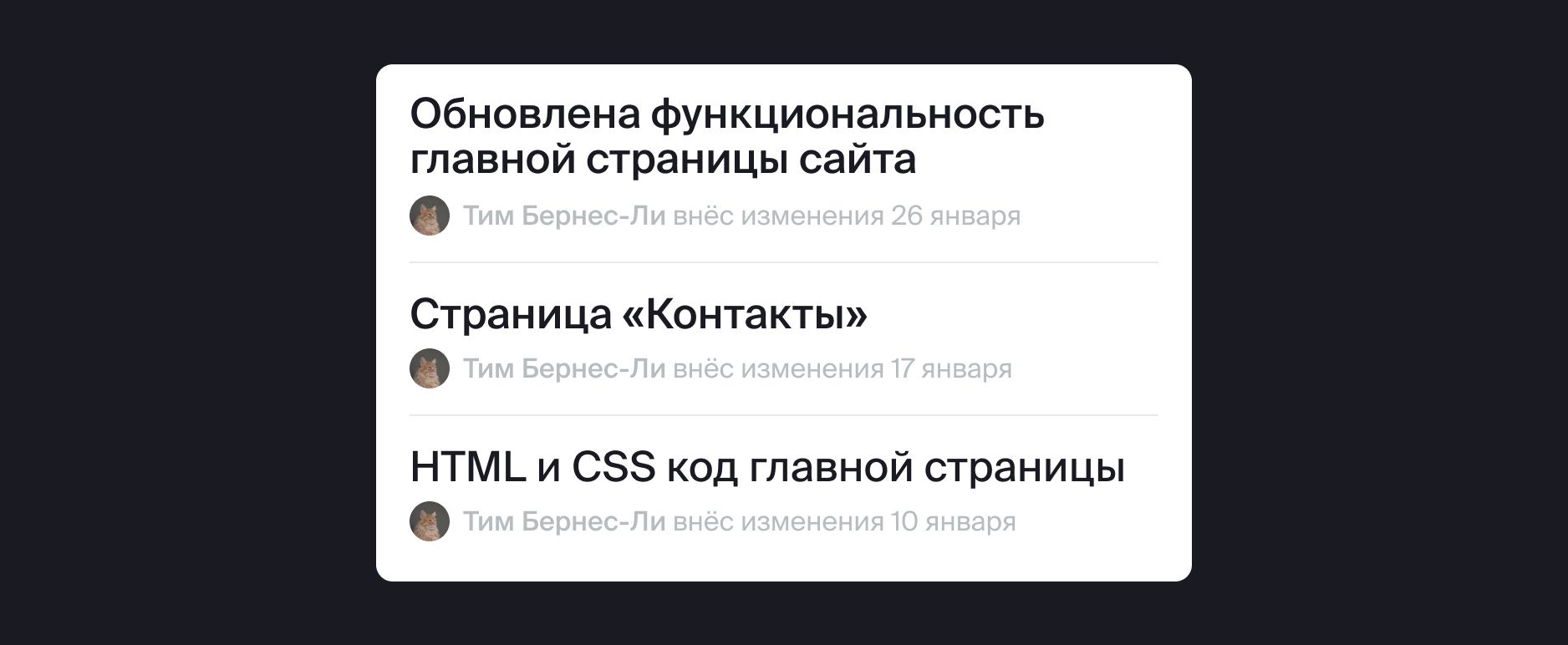
Вы начали знакомиться с вёрсткой, и впереди — первый самостоятельный проект, в котором нужно будет писать много HTML и CSS. Такой проект всё больше похож на реальную работу, поэтому мы постепенно начнём знакомиться с инструментами, которые веб-разработчики используют ежедневно. Первый такой инструмент — Git. Прежде чем разбираться, что это, расскажем, зачем он нужен.



**Контроль версий**

Представьте, вы сделали сайт для заказчика. Сайт работает, люди заходят на него. Заказчик попросил вас переработать часть функциональности. Он думает, что так сайтом будет пользоваться проще. Вы внесли изменения, всё обновили и на время забыли об этом. Через месяц заказчик написал, что его предположение было неверным и он хотел бы вернуть всё как было. Но у вас не сохранилась предыдущая версия файлов и всё приходится переписывать заново.

Этого бы не случилось, если бы версии файлов из разного времени можно было сохранять. Мы бы просто вернулись к предыдущей версии и нам не пришлось делать что-то заново. Представьте, что вся история сохраняется и можно вернуться в код из прошлого:



Нам повезло, «Гит» умеет решать такие задачи. По мере написания и переработки кода его разные версии можно сохранять. Именно поэтому «Гит» называют системой контроля версий.

**Работа в команде**

Вы трудитесь над проектом и задачами в одиночку, но в реальной работе вам предстоит писать код в команде. Одним сайтом могут заниматься несколько человек.

Например, вы делаете сайт вместе с другом. Каждый раз, когда вы вносите изменения в код, вы отправляете другу обновлённый архив. Друг скачивает его, вносит свои изменения и отправляет вам. В следующий раз вы не стали ждать, пока друг завершит свою часть работы, и внесли изменения, ведь работать параллельно быстрее.

В итоге вы одновременно скинули друг другу архивы с изменениями. Но как их совместить? Изменений много, и вы уже толком не помните, что меняли. Придётся открыть ваш код слева, код друга справа и сверять всё вручную. Не тратьте время, «Гит» сделает это автоматически.

**Резюме**

Возможность «сохраняться» и «склеивать» труд разных людей делает «Гит» незаменимым инструментом в командной работе над кодом. Большинство работодателей требуют умения работать с ним. Именно поэтому мы с самого начала интегрируем его в проекты.

**Установка Git**

Как и любую программу, «Гит» необходимо установить. Для разных операционных систем порядок действий различается. Выберите свою и приступайте к установке.

**Windows**

Скачайте и установите «Гит» на свой компьютер. [Ссылка на скачивание с официального сайта Git.](https://git-scm.com/downloads/win)

Когда запустите установщик, вам предложат использовать GitBash — это ещё одна программа для полноценной работы с «Гитом». Обязательно поставьте галочку для её установки. Также поставьте галочку на пункте: «Использовать Git из командной строки» (Use Git... from the Command Prompt). Подробнее об этом в следующем уроке.

**macOS**

Скачайте и установите «Гит» на свой компьютер. [Ссылка на скачивание с официального сайта Git.](https://git-scm.com/downloads/mac)

Тут всё просто, но есть один нюанс: программа-установщик предложит варианты применения «Гита». Выберите «Использовать Git из командной строки» (Use Git ... from the Command Prompt). Подробнее расскажем в следующем уроке.

**Linux**

У Linux существует множество версий. Чтобы установить «Гит», найдите подходящую версию в [разделе установки Git на Linux](https://git-scm.com/downloads/linux).

Для установки нужно использовать терминал. Найдите программу Terminal в поиске или выберите её в списке программ. С терминалом можно взаимодействовать только текстовыми командами. Найдите команду установки «Гита» для своей версии Linux, скопируйте её в терминал и нажмите Enter. Поздравляем, «Гит» установлен.

**Ещё один инструмент — командная строка**

Начало формы

«Гит» установлен. Как начать им пользоваться?

Раз установлен, значит, в меню появилась программа Git. Нужно её открыть, а там в интерфейсе уже разберёмся.

Правильный ответ

Инструменты программистов часто выглядят пугающе. У них как будто нет интерфейса. Программист куда-то вводит текст, и там что-то происходит, как в кино про хакеров. Может, «Гит» — что-то из этой серии?

«Гит» обычно используют через командную строку. Не пугайтесь, хоть в фильмах это и выглядит внушительно, ничего сложного в командной строке нет.

Конец формы

Мы привыкли, что у программ есть графический интерфейс: кнопки, выпадающие меню с настройками. Командная строка — тоже интерфейс, только текстовый. В ней надо вводить команды, чтобы взаимодействовать с программами. Например, обычно, чтобы создать папку, вы делаете правый клик и выбираете нужный пункт в графическом меню. Но папку можно создать и через командную строку, введя соответствующую команду и имя папки. Это просто другой интерфейс.

От слов к делу. Чтобы открыть командную строку:

* на Windows откройте GitBash;
* на Mac OS и Linux откройте программу Терминал (Terminal на английском).

Раз уж мы начали с примера создания папки, разберёмся, как сделать это через командную строку.

Начало формы

Ваше первое действие перед созданием новой папки в обычной жизни?

Задать папке имя. Без имени нельзя.

Тоже правильный ответ

Переместиться в папку, в которой нужно создать новую.

Мы всегда создаём папку где-то в файловой системе. Даже создание папки на рабочем столе подразумевает, что мы переместились в папку «Рабочий стол», когда включили компьютер.

Неправильный ответ

Кликнуть правой кнопкой мыши, чтобы меню открылось.

Конец формы

В графическом интерфейсе всегда ясно, где именно вы находитесь в файловой системе. Если перед вами рабочий стол, значит, вы в папке «Рабочий стол». Если открыта папка «Документы» — вы в ней.

В командной строке вы тоже всегда находитесь в какой-то папке, этого просто не видно. Узнать, где вы сейчас, позволяет команда pwd. Расшифровывается как Print Working Directory (покажи рабочую папку). Введите pwd в командную строку и нажмите Enter:

Скопировать код

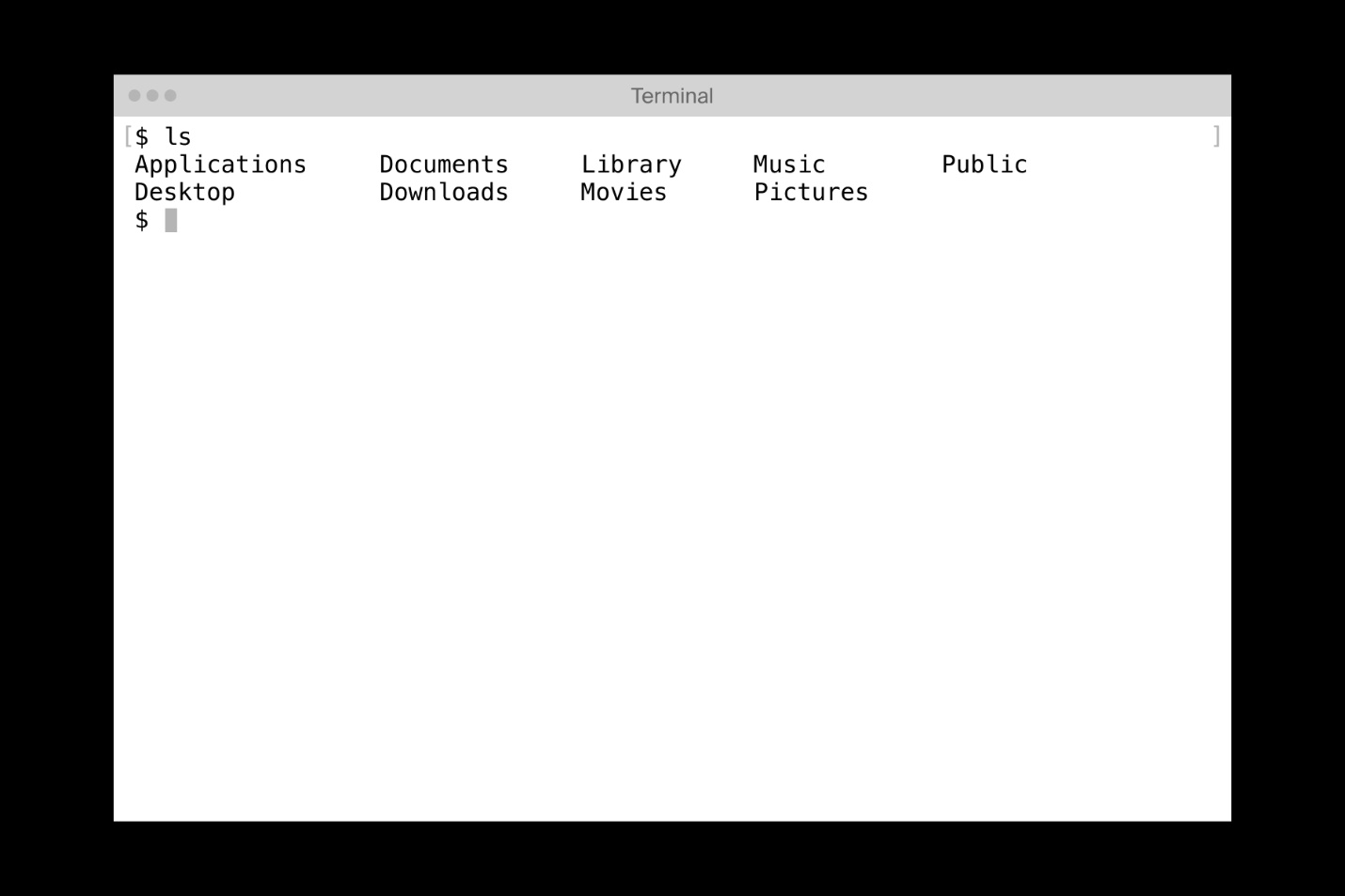
BASH

pwd

/Users/stas\_basov

Командная строка вывела ответ — это путь к папке, в которой мы сейчас находимся. У вас вместо stas\_basov будет имя вашего пользователя. Это домашняя директория — каталог с пользовательскими файлами. Когда вы открываете командную строку, вы сразу оказываетесь в ней.

В графическом интерфейсе вы просто видите содержимое открытой папки. Чтобы сделать это в командной строке, есть команда ls. Это сокращение от list directory contents — «вывести список содержимого каталога». Если ввести эту команду и нажать Enter, вы увидите содержимое текущего каталога — его файлы и папки.



Мы поняли, где находимся, и осмотрелись вокруг, пора научиться ходить. Чтобы переместиться из одной папки в другую, пользуйтесь командой cd. Это сокращение от change directory (сменить каталог)*.* Синтаксис у команды такой: cd имя\_папки:

Скопировать код

BASH

cd Desktop *# переходим на рабочий стол*

pwd

/Users/stas\_basov/Desktop *# теперь мы здесь*

Если в названии папки есть пробелы, используйте кавычки:

Скопировать код

BASH

cd "Рабочий стол"

Чтобы вернуться в родительскую директорию — то есть на уровень выше, вместо названия папки нужно написать две точки: ..:

Скопировать код

BASH

cd .. *# переходим на уровень вверх*

pwd

/Users/stas\_basov *# вернулись назад*

Почему именно две точки? Да кто бы знал.

**Создание папок и файлов**

Пришло время действовать. Команда mkdir создаёт папку. Команде нужно передать имя новой папки.

Создайте в домашней директории папку dev — она станет местом для ваших учебных проектов:

Скопировать код

BASH

mkdir dev *# создаст папку dev в текущем каталоге*

Переместитесь в папку dev, создайте в ней папку first-project и переместитесь в неё:

Скопировать код

BASH

cd dev

mkdir first-project

cd first-project

Создадим нужные файлы. Это делают командой touch, передавая ей имя файла, например:

Скопировать код

BASH

touch index.html *# создаст файл index.html в текущей папке*

Самостоятельно создайте файл style.css.

В конце этого урока в вашей домашней директории должна быть папка dev, в ней папка first-project с двумя файлами: index.html и style.css.

**Удаление папок и файлов**

Напоследок научимся удалять файлы и папки через командную строку. Для удаления файлов используют команду rm, от английского remove, удалять:

Скопировать код

BASH

rm about.html *# удалит about.html из текущей папки*

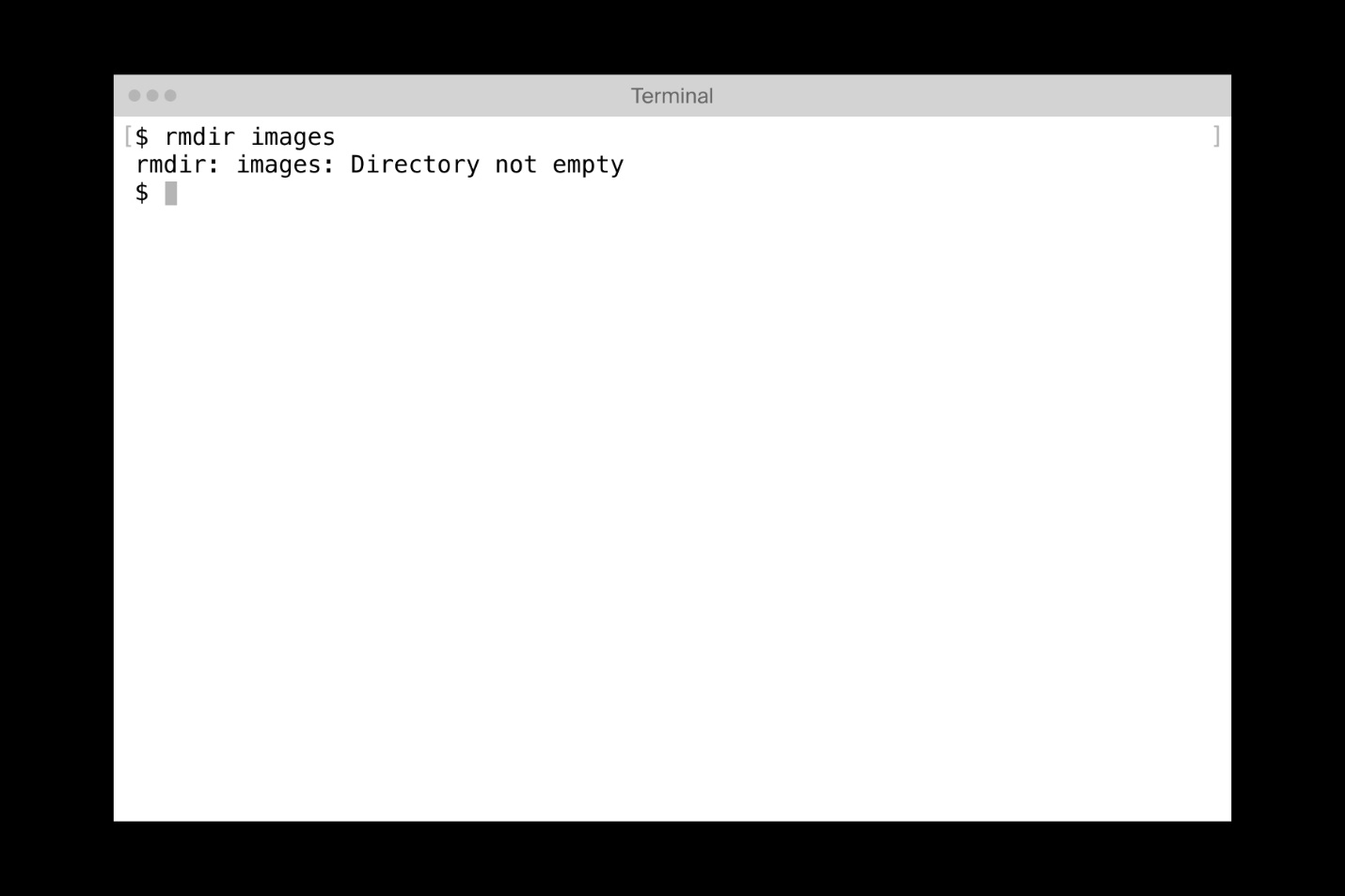
Удалить папку можно командой rmdir:

Скопировать код

BASH

rmdir images *# удалит папку images из текущей папки*

Но если в папке, которую вы пытаетесь удалить, есть какие-то файлы, командная строка не удалит её и выведет сообщение, что папка не пуста:



Это — защита от случайного удаления нужных файлов. Если папку всё-таки нужно удалить, можно использовать команду rm так:

Скопировать код

BASH

rm -r images *# удалит папку images из текущей папки*

В данном случае -r называют ключом. Говорят, «мы вызвали команду rm с ключом r». Этот ключ отвечает за рекурсивное удаление файлов и папок. Это значит, что процесс удаления будет применён для всего содержимого директории. Будьте аккуратны, удаление файлов командами rm и rmdir необратимо — они идут мимо корзины и удаляются навсегда.

**Настройка Git**

Теперь вы джедай командной строки. В этом уроке мы будем использовать её для настройки «Гита». Чтобы другим разработчикам было понятно, кто какие изменения вносил, вам нужно выбрать себе имя — прямо как в компьютерной игре. Давайте сделаем это и начнём использовать «Гит» в полную силу.

**Настройка**

Если работаете на macOS или Linux, запустите программу Терминал. Если на Windows — Git Bash. Пора указать ваше имя и адрес электронной почты. Для этого в командной строке запустите команду git config с опцией --global. Свои данные укажите как значения user.name и user.email:

Скопировать код

BASH

git config --global user.name "Stas Basov"

*# вводите своё имя или ник латиницей и в кавычках*

git config --global user.email "stasbasov@yandex.ru"

*# здесь нужно ввести свой реальный e-mail*

Теперь проверьте, что получилось: выполните команду git config с опцией --list.

Скопировать код

BASH

git config --list

*# вывели в окно командной строки список всех свойств конфига*

В ответ командная строка покажет ваши настройки:

Скопировать код

BASH

user.name=Stas Basov

user.email=stasbasov@yandex.ru

Пользователям macOS и Linux нужно будет выйти из просмотра файла нажатием клавиши Q (сработает при включённой латинской раскладке).

**Где хранятся настройки**

Неважно, в какой директории вы находитесь в командной строке, если нужно просмотреть или изменить настройки «Гита». Вызов git config --global сработает отовсюду.

Вот и всё, первая настройка — самая важная, теперь вы можете использовать «Гит» на полную!

**Подключаем Git к проекту**

Чтобы «Гит» начал работать в проекте, папку с ним нужно сделать git-репозиторием. Это значит, что «Гит» начнёт отслеживать все изменения внутри этой директории. Для этого переместитесь в папку с проектом и введите команду:

Скопировать код

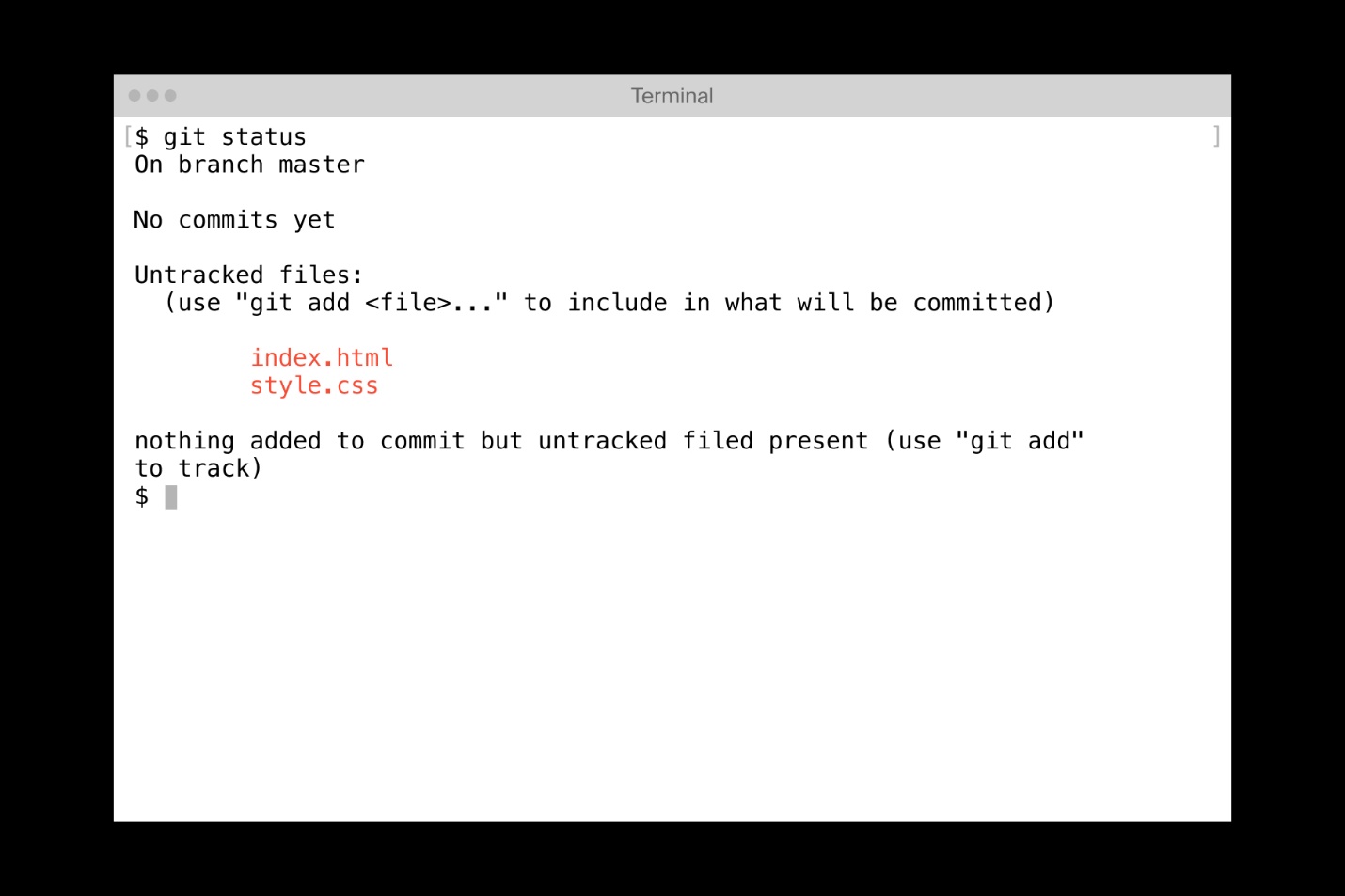
BASH

git init

Командная строка сообщит, что репозиторий инициализирован.

Сделайте git-репозиторием папку first-project. Для этого перейдите в папку first-project командой cd и напишите в терминал git init.

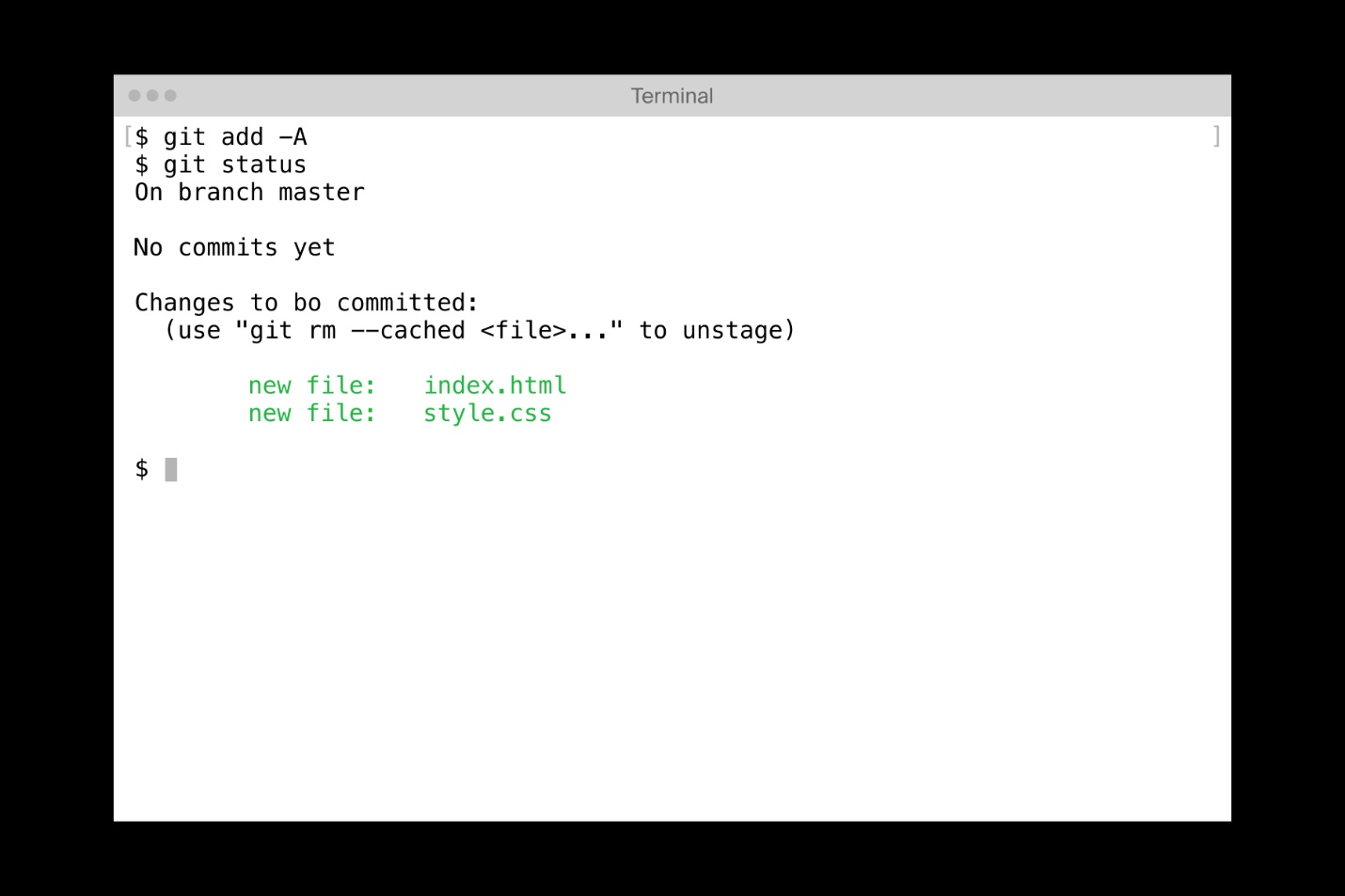
Запустите команду git status. «Гит» сообщит, что в этой папке есть два файла: index.html и style.css:



«Гит» называет эти файлы ”Untracked files“ — неотслеживаемые файлы. Это значит, что «Гит» их видит, но, если попытаться сохранить их версию сейчас, «Гит» этого не сделает.

Это не то, что мы хотели, наша цель — научиться сохранять версии. Для этого файлы нужно подготовить к сохранению. За это отвечает команда git add. При её использовании нужно указать имя файла, судьбу которого мы хотим зафиксировать в текущем виде, например: git add index.html.

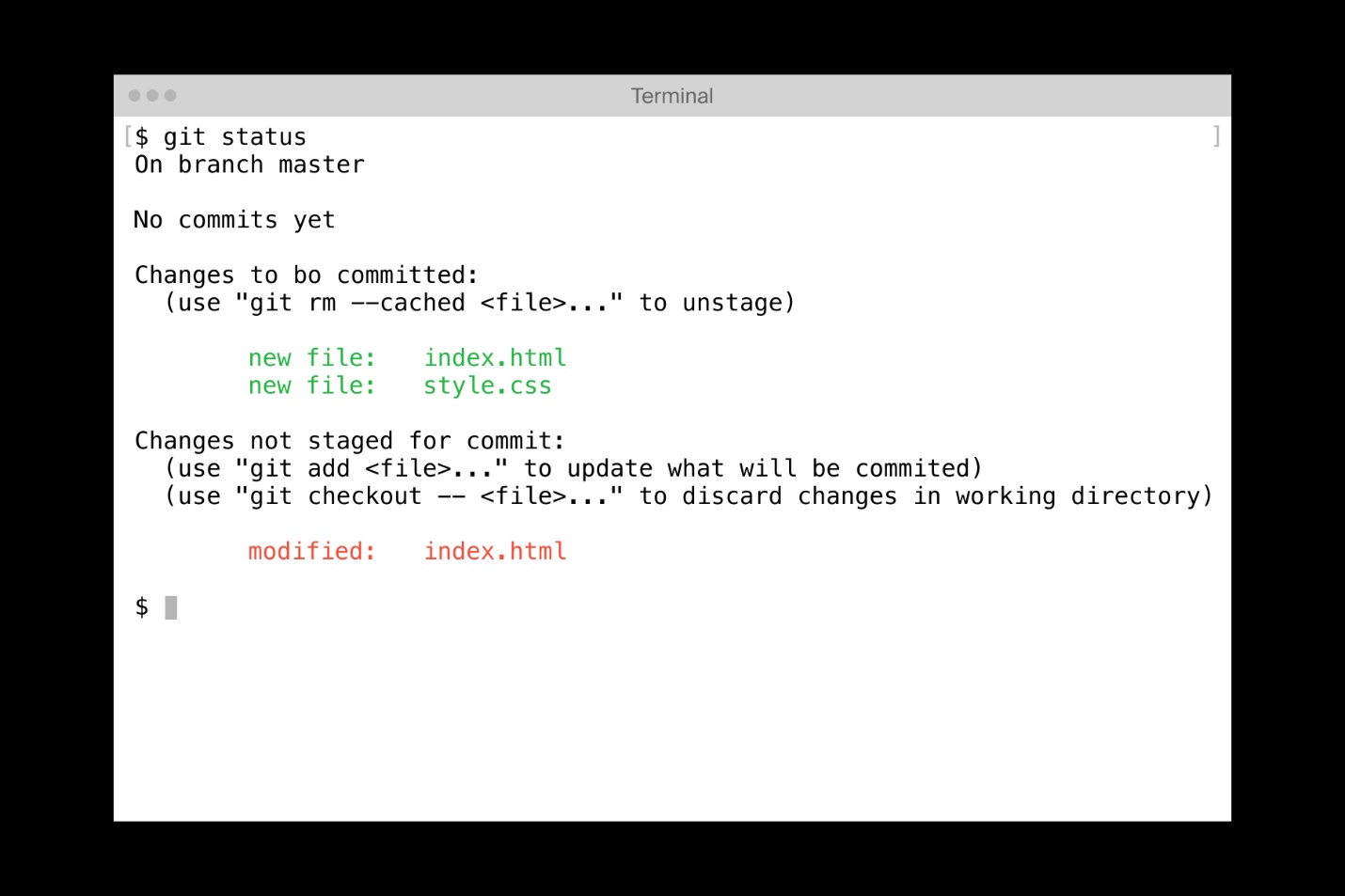
Если мы хотим сохранить состояние всех файлов, можно прибегнуть к опции --all. git add --all или короче git add -A:



Файлы, отмеченные зелёным, готовы к сохранению в их текущих состояниях. Именно об этом команда git add сообщила «Гиту». Но сохранения пока не произошло: сначала нужно сообщить «Гиту», чтó мы хотим сохранить, и только потом сохранять.

Сохранение версии файлов в «Гите» называют «коммитом». Сделать коммит — значит сохранить текущее состояние файлов. Этим мы займёмся в следующем уроке.

Если сейчас отредактировать любой из файлов, он снова загорится красным:



Чтобы в коммит ушла его обновлённая версия, нужно снова использовать git add так:

Скопировать код

BASH

git add index.html

Или использовать сокращённую версию git add --all:

Скопировать код

BASH

git add -A

Это подготовит к сохранению все изменённые файлы.

Начало формы

Какой командой инициализировать «Гит» в проекте?

Правильный ответ

git init

Да, верно: если перейти в каталог с проектом и выполнить команду git init, «Гит» начнёт отслеживать файлы в выбранной директории.

Конец формы

Начало формы

Какая команда говорит «Гиту» подготовить все файлы к сохранению?

Правильный ответ

git add --all

Верно — add --all означает «подготовить все файлы».

Тоже правильный ответ

git add -A

Правильно: это укороченная версия команды git add --all.

Конец формы