Разделы

- CMS для сайта
- Языки программирования
- Технологии, инструменты, сервисы
- Хостинг, домены и серверы
- Видео уроки
- Новости и статьи
- SEO
- Часто задаваемые вопросы

Поделиться с друзьями

Git команды

В данной статье будут собраны основные Git команды, в том числе команды, которые используются в командной разработке. Эту статью можно использовать в качестве шпаргалки при работе с Git.

Основные Git команды.

Для начала нужно скачать git, это можно сделать, по этому url: https://gitscm.com/downloads

Если вы только установили Git, то нужно будет настроить 2 переменные: user.name и user.email - это подписи коммитов. По ним будет ясно, кто вносил правки в код. Не путайте эти настройки с настройками, которые вы указывали на Github, они не имеют никакого к ним отношения.

Чтобы указать имя пользователя, нужно ввести следующую Git команду:

```
git config –global user.name "Name LastName"
```

Чтобы указать Email пользователя, нужно ввести Git команду:

```
git config –global user. email user@mail.ru
```

Проверить текущие настройки можно Git командой:

```
git config –list
```

Затем, когда уже создан проект, и написаны первые строки кода, нужно инициализировать проект. Это делается один раз, но важно не забывать это делать - как помыть руки перед едой.

Инициализировать проект. Используется Git команда:

```
git init
```

Затем нужно **проиндексировать** перед коммитом созданные файлы и изменения

```
git add index.html
```

Если создано много файлов или внесено много изменений к ним, то тогда можно использовать следующую Git команду:

```
git add .
```

Данная команда добавит в индекс все файлы, в которых были изменения, а также новые файлы.

После того, как изменения были проиндексированы, их нужно **закоммитить**:

```
git commit –m "Название коммита"
```

Также можно сократить эти две записи:

```
git commit –am "Название коммита"
```

Эта запись одновременно проиндексирует файлы (за которыми ведется слежка).И закоммитит их.

После создании нового репозитория на github, он вам сгенерирует команды. Их нужно будет только скопировать и вставить в консоль

Вот, как они будут выглядеть:

```
git remote add origin https://github.com/YouName/test.git
```

```
git push -u origin master
```

Данные Git команды нужно ввести 1раз. После это го нужно будет указывать только git push, чтобы загрузить данные на Ваш удаленный репозиторий

Ветвления

Создание ветки:

```
git branch Имя_ветки
```

Переключение между ветками:

```
git checkout Имя_ветки
```

Создание ветки и переключение на нее:

```
git checkout –b Имя_ветки
```

Переименование ветки:

```
git branch –m Старое_название_ветки Новое_название_ветки
```

Удаление ветки:

```
git branch –d Имя_ветки
```

Слияние веток

Алгоритм:

1.Переключиться на ветку, в которой будут приняты изменения:

```
git checkout Имя_ветки
```

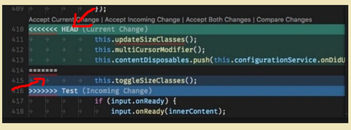
2. Указать ветку, в которой будут приняты изменения:

```
git merge ветка_с_нужными_изменениями
```

Конфликты

Для того, чтобы выяснить, в каких файлах есть конфликты, нужно набрать Git команду:

```
git status
```



Все, что находится в HEAD - это наши изменения, и все, что находится после знаков равно - это внешние изменения. Вам нужно выбрать, какие изменения оставить.

После того как решите конфликты, нужно использовать Git команду:

```
git commit
```

Клонировать удаленный репозиторий:

```
git clone git://myrepo.com/project.git
```

Данная команда клонирует полностью проект.

Клонировать удаленный репозиторий в определенную папку

```
git clone git://myrepo.com/project.git mydir
```

Стянуть изменения:

```
git pull
```

Стянуть определённую ветку с удаленного репозитория:

```
git pull origin Name_branch
```

Отмены

Отменить индексацию файла можно так:

```
git reset HEAD имя файла
```

Если вы уже внесли изменения в файл и сохранились, но хотите **отменить изменения в файле**, то можете указать:

```
git checkout – Имя файла
```

И он вернет состояние файла предыдущего сохранения

Редактирование последнего коммита:

```
git commit –amend -m "Комментарий к коммиту"
```

Отменить процесс слияния:

```
git merge –abort
```

Поделиться с друзьями

Была ли данная статья полезна?

Статьи раздела Комментарии

- Лучшие редакторы кода для верстальщиков
- Топ 4 лучших PHP редакторов
- Лучшие редакторы для Python
- Полезные плагины для Brackets
- Sublime Text полезные плагины
- 10 лучших визуальных редакторов WYSIWYG
- 17 Самых полезных расширений для разработчиков в Chrome
- 18 Бесплатных адаптивных слайдеров на jQuery
- 16 Самых полезных инструментов для веб-разработчиков.
- Обзор редактора HTML Source
- Загрузка страницы с зарисовкой. Анимация SVG
- NVU обзор HTML редактора(программы)
- Создание логотипа средствами CSS3
- Подробный обзор mytaskhelper
- vs code плагины
- CodeLobster - обзор IDE