Бесплатная электронная книга

# УЧУСЬ

# github

Free unaffiliated eBook created from
**Stack Overflow contributors.**

#github

# Около

You can share this PDF with anyone you feel could benefit from it, downloaded the latest version from: github

It is an unofficial and free github ebook created for educational purposes. All the content is extracted from Stack Overflow Documentation, which is written by many hardworking individuals at Stack Overflow. It is neither affiliated with Stack Overflow nor official github.



Use the content presented in this book at your own risk; it is not guaranteed to be correct nor accurate, please send your feedback and corrections to info@zzzprojects.com

# глава 1: Начало работы с github

## замечания

В этом разделе представлен обзор того, что такое github, и почему разработчик может захотеть его использовать.

Следует также упомянуть о любых крупных предметах в рамках github и ссылки на связанные темы. Поскольку документация для github является новой, вам может потребоваться создать начальные версии этих связанных тем.

## Examples

### Установка или настройка

GitHub - огромная коллекция репозиториев Git. Другими словами, вы можете думать о GitHub как о наборе многих проектов!

# Создание учетной записи

- Посетите главную страницу GitHub в этом
- Выберите имя пользователя, введите свой адрес электронной почты и выберите безопасный пароль, и вы готовы к работе!

# Полезные инструменты

Для начинающих Git / GitHub понимание того, как работает управление версиями, может сначала ввести в заблуждение. Существует версия GUI GitHub, которую вы можете скачать и использовать. GitHub Desktop - это тот инструмент.

# Создание первого хранилища

Вы можете думать о репозитории как о проекте. Вы можете создать репозиторий онлайн или офлайн. Выполните следующие действия:

## онлайн

1. Сначала войдите в систему и перейдите в свой профиль.

2. Перейдите на вкладку «Хранилища» в верхней части страницы.
3. Нажмите зеленую кнопку «Создать», и вы готовы гулнуть!

## Не в сети

1. Загрузите и установите git (выберите операционную систему, в которой вы работаете)
2. После загрузки и установки вы можете использовать инструмент командной строки или загрузить клиент GUI.
3. После установки создайте учетную запись на github
4. В правом верхнем углу нажмите + и выберите либо создание нового репозитория, либо импорт существующего.
5. Если вы выберете новую, введите имя репозитория и выберите либо общедоступную, либо приватную.
6. Нажмите: Создать репозиторий

NB Частные репозитории недоступны для бесплатных пользователей.

### Файл README

Если ваш проект не имеет README.md, GitHub может анализировать README.rdoc для отображения деталей. Если он есть, он будет использовать README.md, молча игнорируя rdoc.

---

# Файл README может включать-

## Название Проекта

Опишите кратко о своем проекте. Вы также можете предоставить ссылку на веб-сайт проекта, значки, информацию об сообществах и контактах (например, электронную почту, социальный сайт).

## Скачать

Ссылка на исполняемый файл (исполняемый файл или миниатюрный или установочный файл). Также могут быть ссылки на предыдущие версии.

## Монтаж

Как ваша работа может быть использована. Он может включать предварительные

условия, настройки, сторонние библиотеки, использование, предостережения и т. Д.

## демонстрация

Он может включать в себя образец кода, файл gif, ссылку на видео или даже скриншоты.

## Авторы

Имена авторов, контактная информация и т. Д.

## Подтверждения

Список людей или сообщества помог и вдохновлен во всем проекте

## Содействие

Инструкции по внесению вклада (т. Е. Добавить функцию, сообщить об ошибке, отправить патч) в проект. Может также содержать ссылку на документацию.

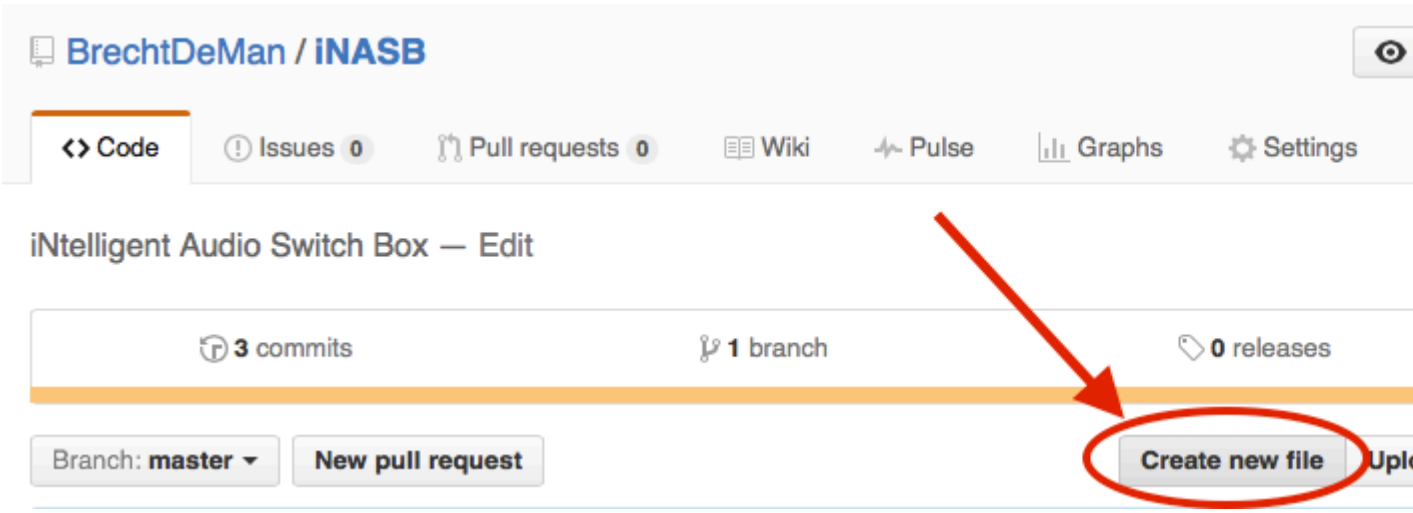
## Лицензия

Дайте краткое введение по вашей лицензии. Вы также можете указать ссылку на сайт лицензии.

### ЛИЦЕНЗИОННЫЙ файл

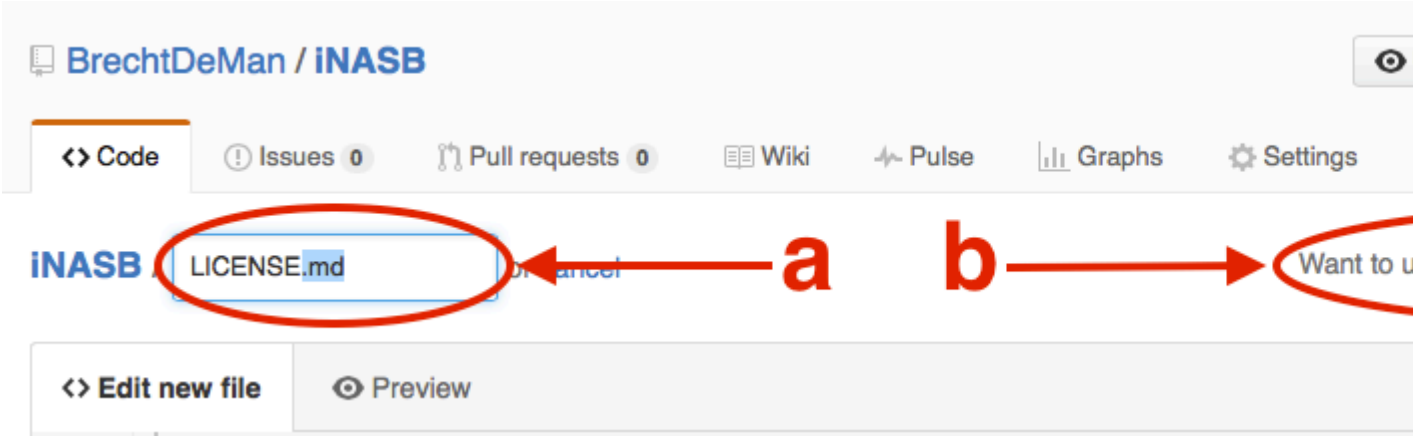
GitHub помогает вам быстро добавить лицензию в ваш репозиторий, в качестве альтернативы для добавления собственного файла text / markdown.

1. В вашем репозитории нажмите «Создать новый файл»,

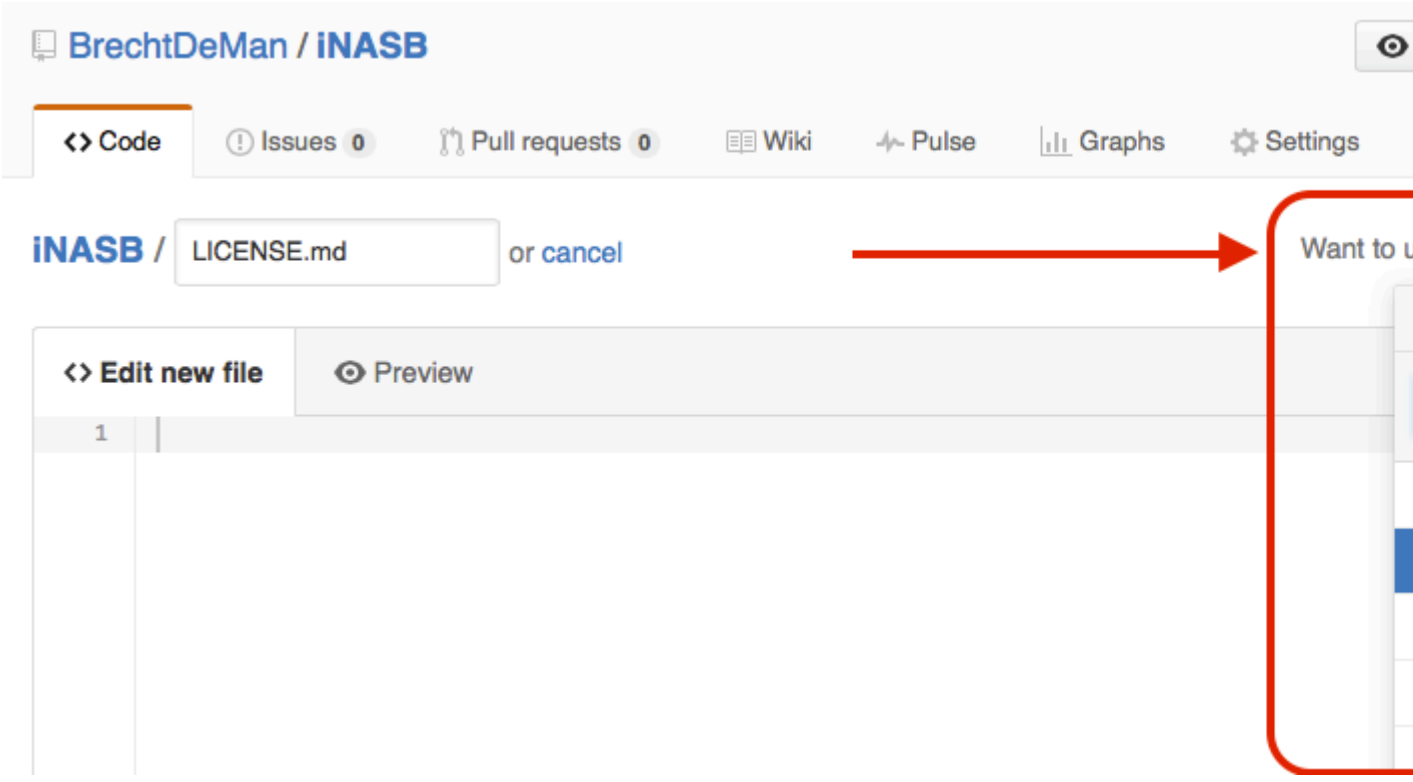


2. На следующей странице:

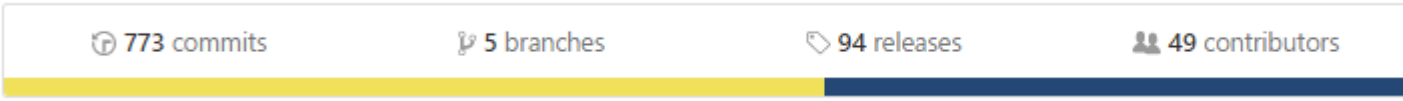
    1. Введите имя файла `LICENSE.md` или `LICENSE.txt` качестве имени файла нового файла.
    2. *Хотите использовать новый шаблон?* появится диалоговое окно.




3. Выберите предпочтительную лицензию.




4. Лицензию, которую вы можете увидеть в репозитории:




*Из вопросов и ответов - Как добавить лицензию на существующий проект Github*

## GitHub Flavored Markdown

GitHub расширяет синтаксис Markdown для предоставления новых полезных функций.

## заголовок

```
# Header1
## Header2
### Header3
#### Header4
##### Header5
###### Header6
H1
===
H2
---
```

# Header1

## Header2

### Header3

#### Header4

##### Header5

###### Header6

# H1

## H2

## акцент

```
*Italic1* _Italic2_
**Bold1** __Bold2__
***Bold_Italic***
~~Strikethrough~~
```

*Italic1 Italic2*
**Bold1 Bold2**
***Bold_Italic***
~~Strikethrough~~

# Горизонтальная линия

```
---
***
___
```

---

---

---

# Список

```
unordered list:

* item-1
  * sub-item-1
  * sub-item-2
- item-2
  - sub-item-3
  - sub-item-4
+ item-3
  + sub-item-5
  + sub-item-6


ordered list:

1. item-1
  1. sub-item-1
  2. sub-item-2
2. item-2
  1. sub-item-3
  2. sub-item-4
3. item-3
```

unordered list:

- item-1
  - sub-item-1
  - sub-item-2
- item-2
  - sub-item-3
  - sub-item-4
- item-3
  - sub-item-5
  - sub-item-6

ordered list:

1. item-1
   i. sub-item-1
   ii. sub-item-2
2. item-2
   i. sub-item-3
   ii. sub-item-4
3. item-3

# Таблица

```
Table Header-1 | Table Header-2 | Table Header-3
:--- | :---: | ---:
Table Data-1 | Table Data-2 | Table Data-3
TD-4 | Td-5 | TD-6
Table Data-7 | Table Data-8 | Table Data-9
```

| Table Header-1 | Table Header-2 | Table Header-3 |
|:---|:---:|---:|
| Table Data-1 | Table Data-2 | Table Data-3 |
| TD-4 | Td-5 | TD-6 |
| Table Data-7 | Table Data-8 | Table Data-9 |

# Код

```
inline code- `int i=0`

block code-
``` C
for(int i=0; i<10; i++){
    printf("Hallow World! \n");
}
```
```

inline code- `int i=0`

block code-

```C
for(int i=0; i<10; i++){
    printf("Hallow World! \n");
}
```

# котировка

```
> Stay hungry; stay foolish.
>> Quality is better than quantity.
>>> Life is not fair; get used to it.
```

> Stay hungry; stay foolish.
>
> > Quality is better than quantity.
> >
> > > Life is not fair; get used to it.

# Ссылка на сайт

```
https://github.com
[GitHub](https://github.com)
[GitHub](https://github.com "github website")
[GitHub][1]

[1]: https://github.com
```

https://github.com
GitHub
GitHub
GitHub

# Образ

```
![GitHub Logo](https://assets-cdn.github.com/images/icons/emoji/octocat.png "GitHub")
```

# Списки задач

```
- [x] completed item
- [ ] incomplete item
```

- [x] completed item
- [ ] incomplete item

# Emoji

```
:octocat: :+1: :book: :ghost: :bulb: :imp:
```

🐙👍📖👻💡😈

Для всех посетителей GitHub emojies - Emoji Cheat Sheet .

# Ссылки SHA

Любая ссылка на хэш SHA1 коммита будет преобразована в ссылку на сам commit в GitHub:

```
e7909ea4fbb162db3f7f543d43c30684a3fb745f
```

Write Preview

`e7909ea`

# Вызов запроса и выдачи

Любая ссылка на запрос на перенос или проблему будет автоматически связана с этим

запросом на перенос или проблемой.

Это можно сделать, поставив # перед номером вопроса / Pull Request.

Прочитайте Начало работы с github онлайн: https://riptutorial.com/ru/github/topic/1214/начало-работы-с-github

# глава 2: вопросы

## Examples

### Создание проблемы

1. Перейдите на страницу GitHub для проекта, где вы хотите создать проблему.
2. Нажмите « ***Проблемы»*** .
3. В правом верхнем углу нажмите « ***Новый выпуск»*** .
4. Введите название проблемы.
5. Введите тело проблемы (включая журналы, фрагменты кода и т. Д.).
6. *Необязательно:* Чтобы просмотреть проблему перед ее отправкой, нажмите на предварительный просмотр.
7. Нажмите « ***Отправить новую тему»*** .

Прочитайте вопросы онлайн: https://riptutorial.com/ru/github/topic/3757/вопросы

# глава 3: Вытянуть запросы

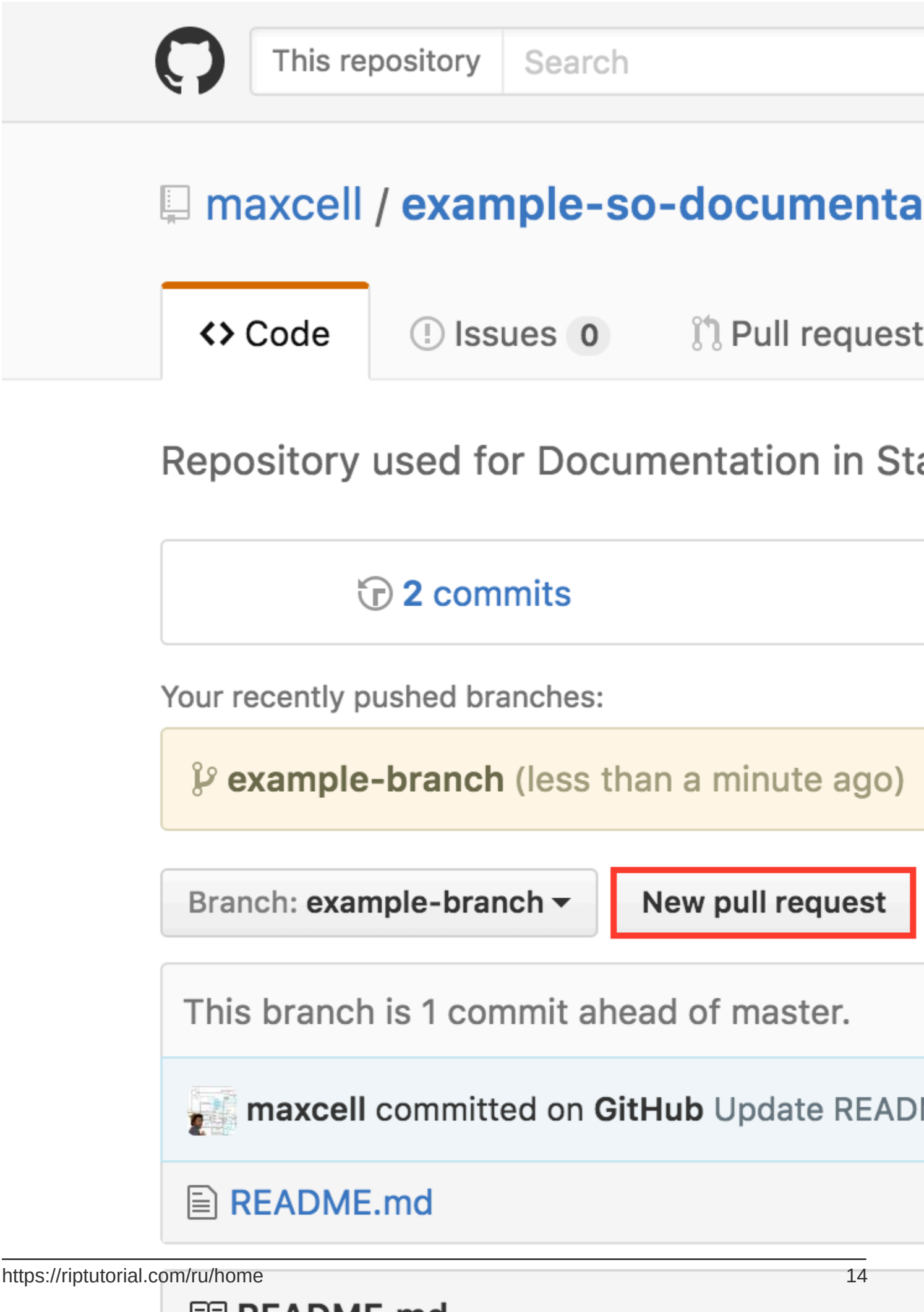
## Examples

### Открытие запроса на растяжение

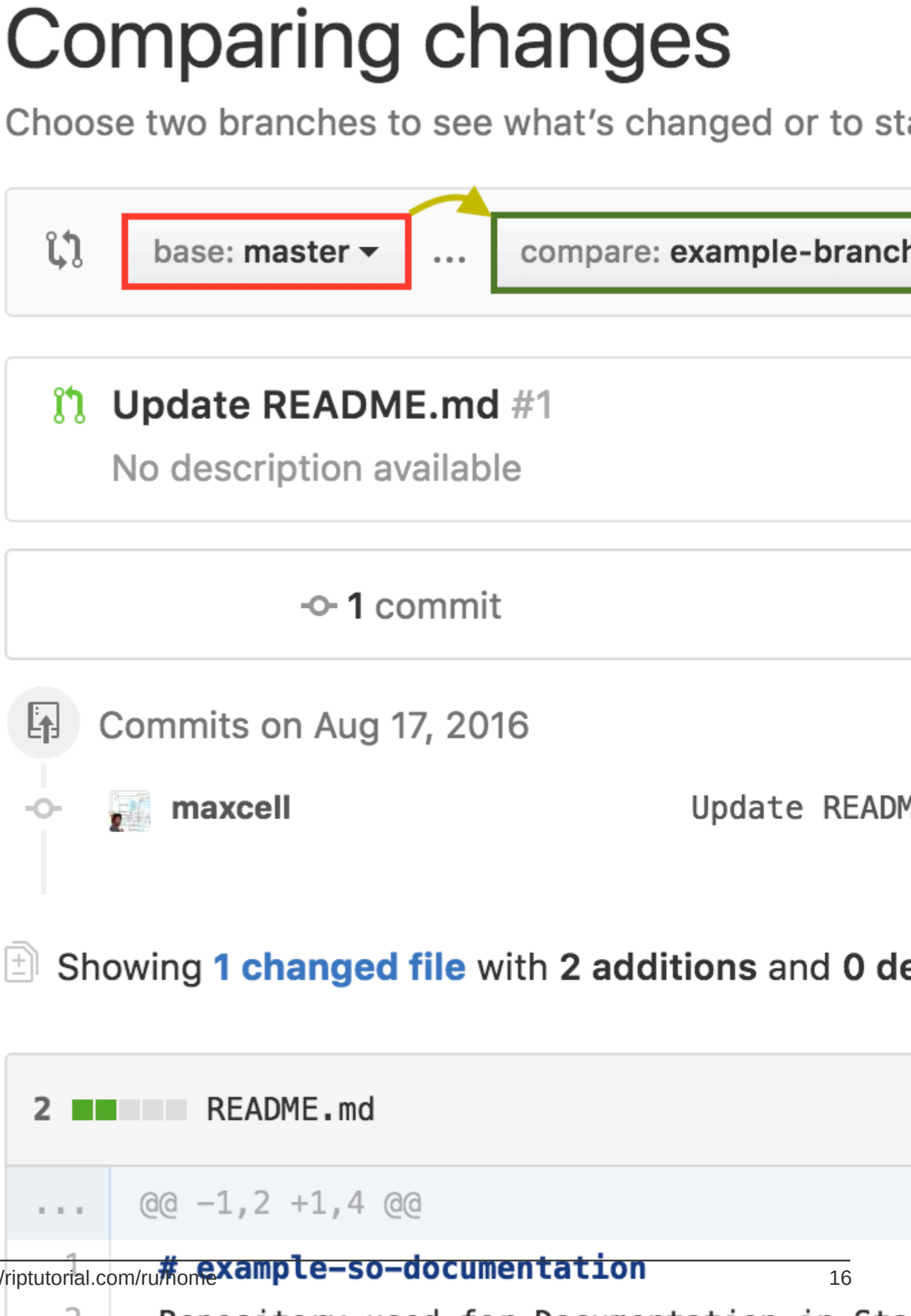
## Новый запрос Pull

Всякий раз, когда вы хотите создать запрос Pull (скажем, это либо недавнее изменение, но вы также можете сделать это со старым изменением!), Вы можете продолжить и позволить GitHub сделать много тяжелой работы для вас и нажмите зеленую кнопку « **Запрос и запрос на** выбор» ( *НЕ ДОЛЖЕН БЫТЬ ЗАВЕРШЕН С КЛОНОМ ИЛИ ЗАГРУЗКОЙ* ) в окне предупреждения, в котором упоминается, что вы просто нажали на ветку.

В противном случае вы будете использовать кнопку **New Pull Request** , расположенную рядом с вашей веткой.


This repository
Search
maxcell / example-so-documenta
Code
Issues 0
Pull request
Repository used for Documentation in Sta
2 commits
Your recently pushed branches:
example-branch (less than a minute ago)
Branch: example-branch
New pull request
This branch is 1 commit ahead of master.
maxcell committed on GitHub Update READ
README.md

немного отличается только потому, что у меня уже есть один открытый.) Вы увидите ниже этого изображения, что он попросит вас **создать запрос Pull** . Вы просто хотите быть уверены, что у вас есть правильная ветка, в которую вы хотите объединиться. BASE означает, что мы хотим изменить и сравнить, к чему мы это будем стремимся! Итак, в этом случае `master` находится за `example-branch` и мы хотим, чтобы `master` был схвачен.


Comparing changes
Choose two branches to see what's changed or to st
base: master
...
compare: example-branch
Update README.md #1
No description available
1 commit
Commits on Aug 17, 2016
maxcell
Update READM
Showing 1 changed file with 2 additions and 0 de
2
README.md
...
@@ -1,2 +1,4 @@
1
# example-so-documentation

вытянуть-запросы

# глава 4: загрузить один файл из репозитория GitHub

## Examples

### из общего хранилища с использованием командной строки и переименования файла

Этот пример захватывает файл Node.gitignore из репозитория gitignore GitHub, загружает его в ваш текущий рабочий каталог и переименовывает его в .gitignore - все очень типичные действия для кого-то, запускающего новый проект node.js.

```
$ curl http://github.com/github/gitignore/raw/master/Node.gitignore -o .gitignore
```

### найти URL-адрес файла, который вы хотите скачать

1. перейдите к нужному файлу в репозитории
2. нажмите кнопку «raw»
3. скопировать URL из адресной строки

См. Следующий пример из репозитория gitignore от GitHub:
http://github.com/github/gitignore/raw/master/Node.gitignore

Вы можете быстро распознать URL-адрес, который будет работать для загрузки отдельного файла или загрузки html-страницы. Найдите подкаталог / raw / right перед именем ветки.

Прочитайте загрузить один файл из репозитория GitHub онлайн:
https://riptutorial.com/ru/github/topic/10898/загрузить-один-файл-из-репозитория-github

# глава 5: Использование Gist

## Вступление

Гисты - отличный способ поделиться своей работой. Вы можете делиться отдельными файлами, частями файлов или полными приложениями. Вы можете получить доступ к gists на странице https://gist.github.com .

Каждый из них представляет собой хранилище Git, что означает, что он может быть раздвоен и клонирован. Редактор gist работает от CodeMirror.

Существует два типа сущностей: публичные сущности и секретные сущности.

Кроме того, если вы не вошли в GitHub, когда будете создавать свой стиль, это будет анонимным.

## замечания

Гисты - отличный способ поделиться своей работой. Вы можете делиться отдельными файлами, частями файлов или полными приложениями.

Существует два типа сущностей: публичные сущности и секретные сущности. Кроме того, если вы не вошли в GitHub, когда будете создавать свой стиль, это будет анонимным.

**Общественные листы**

В Discover открываются публичные узлы, где люди могут просматривать новые gists по мере их создания. Они также доступны для поиска, поэтому вы можете использовать их, если хотите, чтобы другие люди находили и видели вашу работу.

**Секретные Гисты**

Секретные gist не отображаются в Discover и не доступны для поиска. Используйте их, чтобы записать идею, которая пришла к вам во сне, создать список дел или подготовить код или прозу, которые не готовы к общению с миром.

Вы можете создать столько тайных сущностей, сколько захотите.

**Анонимные пользователи**

Если вы создадите gist без входа в GitHub, это будет анонимным сущностью. Анонимные gists могут быть публичными или секретными. Чтобы удалить анонимный текст на GitHub.com или GitHub Enterprise, обратитесь в службу поддержки GitHub или к администратору вашего сайта. Укажите адрес, который вы хотите удалить.

# Examples

## Public Gist

Общественный смысл может быть *почти* всем.

Простой пример функции Javascript:

```
function randomInt(min, max) {
return Math.floor((max - min + 1) * Math.random()) + min;
}
```

## Секретный

Секретный смысл следует использовать для всего, что вы не хотите публично публиковать на GitHub. Секретные сущности могут использоваться, когда вы не хотите, чтобы закрытые ключи были доступны для публики, или для частного кода в целом.

Простой пример кода JSON, который лучше подходит для секретной сущности:

```
{
"id": AKIAIOSFODNN7EXAMPLE,
"secret": wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
}
```

Прочитайте Использование Gist онлайн: https://riptutorial.com/ru/github/topic/7978/использование-gist

# глава 6: Использование кнопок GitHub

## Вступление

Что такое кнопки GitHub? Кнопки GitHub - это кнопки, которые вы можете добавить на свой сайт, который перенаправляет пользователей в любой репозиторий, который вам нравится!

## замечания

Кредиты:

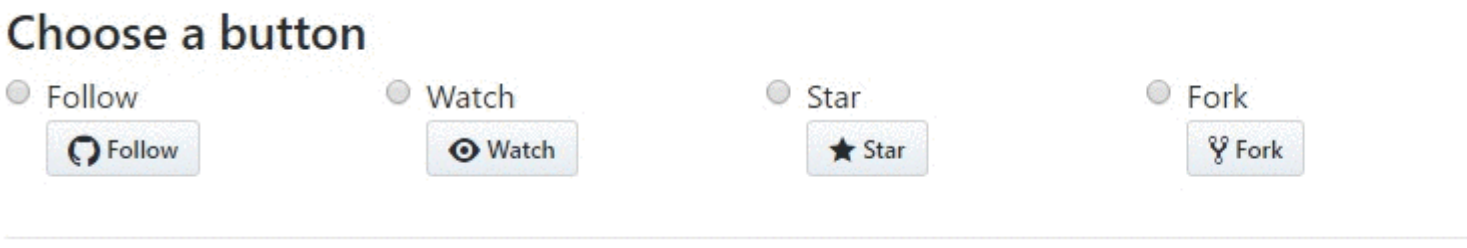
- Изображения Gif, записанные с помощью Recordit
- Статические изображения, снятые с помощью инструмента Snipping Tool
- Редактором кода, используемым в полном учебнике, был codepen.io

## Examples

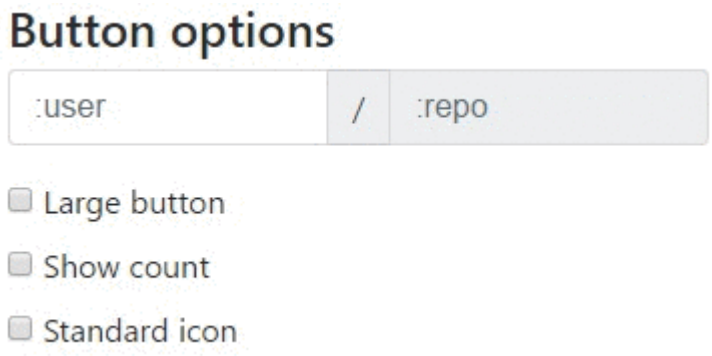
### Следующая кнопка

Кнопка «Далее» - это кнопка, которая ссылается на страницу пользователя GitHub и предлагает пользователю следовать за пользователем. Вот как это сделать:

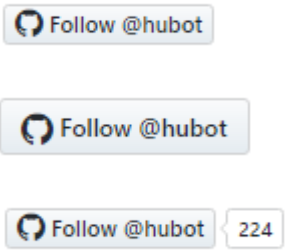
1. Перейти на github: кнопки
2. Нажмите «Follow»




3. Поместите свое имя пользователя GitHub в поле с надписью «: user»



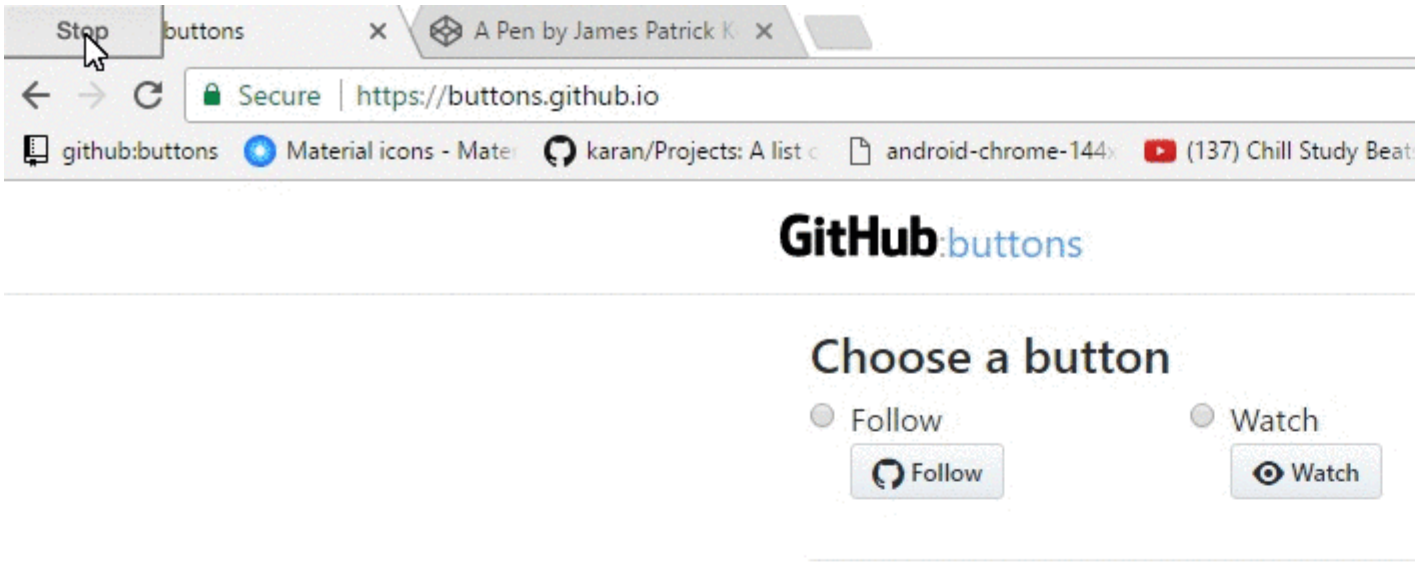

4. Настройте кнопку, используя поля «Большая кнопка», «Показать счет» и «Стандартный значок»:




5. Поместите этот код в `<head>` или до конца `<body>` вашего кода:

```
<a class="github-button" href="https://github.com/hubot" aria-label="Follow @hubot on
GitHub">Follow @hubot</a>
```

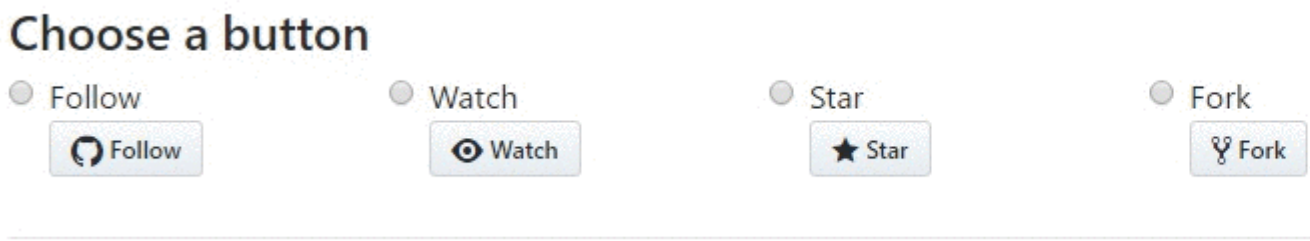
6. Поместите настроенный код рендеринга кнопки в свой код.


Stop
buttons
A Pen by James Patrick K
Secure
https://buttons.github.io
github:buttons
Material icons - Mate
karan/Projects: A list
android-chrome-144
(137) Chill Study Beat
GitHub:buttons
Choose a button
Follow
Follow
Watch
Watch

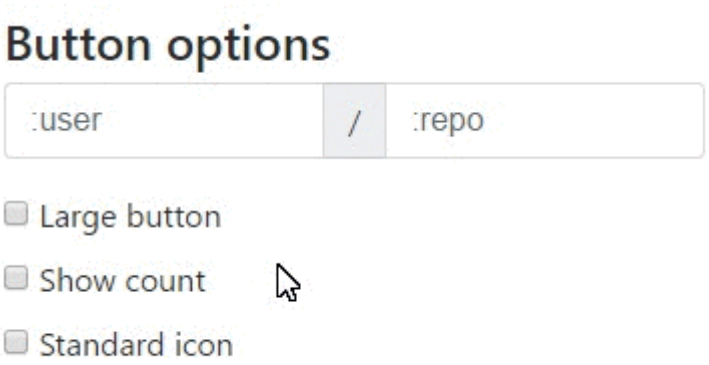
- **Звезда** репозитория
- **Вилка** репозитория
- **Загрузить** репозиторий
- Список **проблем** с репозиторием

Вот как это сделать:

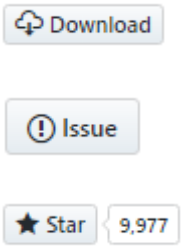
1. Перейти на github: кнопки
2. Нажмите тип кнопки, которую вы хотите создать (Watch, Star, Fork, Download или Issue)




3. Поместите свое имя пользователя GitHub в поле с надписью «: user» и ваш репозиторий в поле «: repo»,



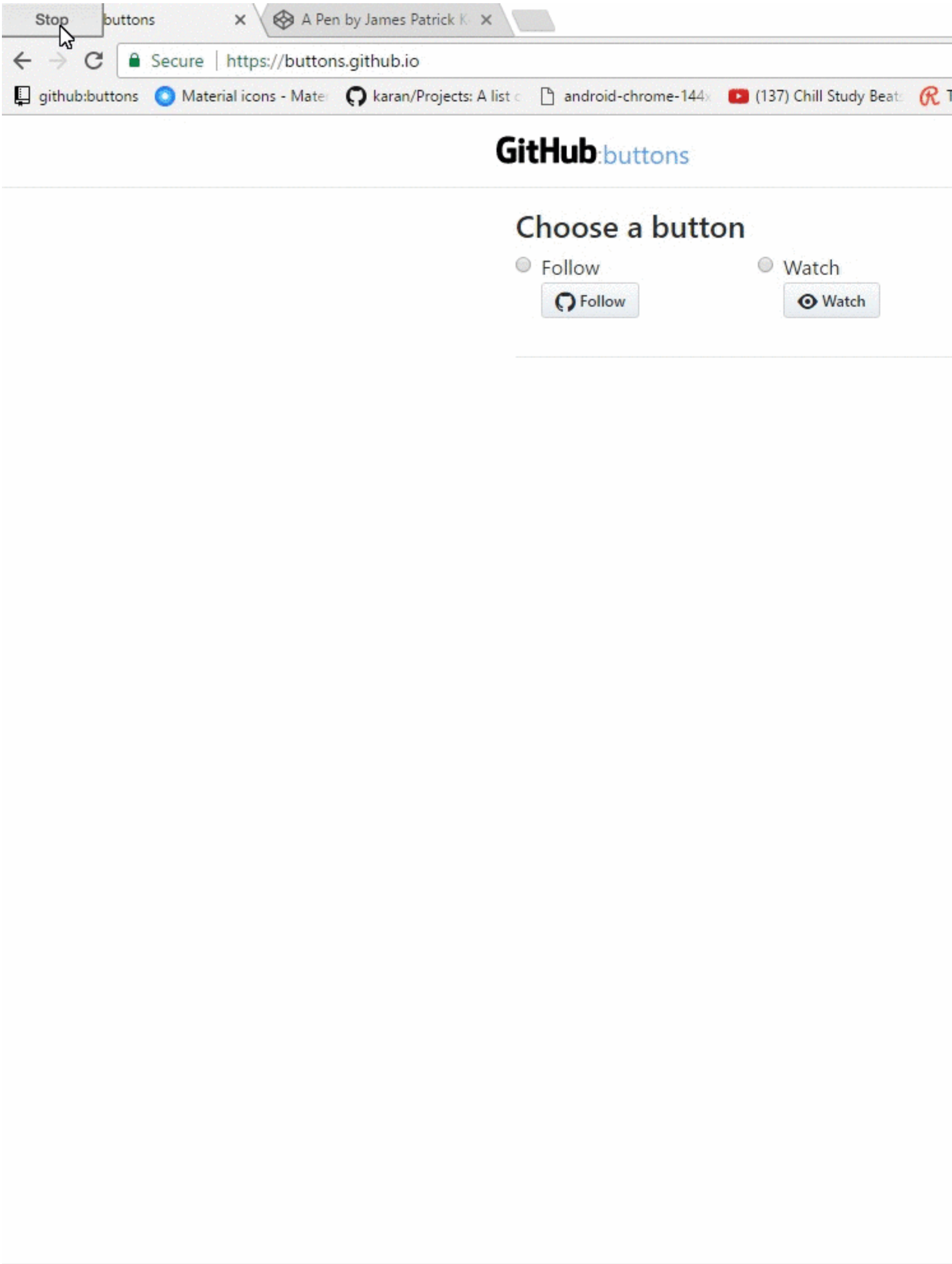

4. Настройте кнопку, используя поля «Большая кнопка», «Показать счет» и «Стандартный значок»:




5. Поместите этот код в `<head>` или до конца `<body>` вашего кода:

```
<a class="github-button" href="https://github.com/hubot" aria-label="Follow @hubot on GitHub">Follow @hubot</a>
```

6. Поместите настроенный код рендеринга кнопки в свой код.


Stop
buttons
A Pen by James Patrick K
Secure
https://buttons.github.io
github:buttons
Material icons - Mate
karan/Projects: A list
android-chrome-144x
(137) Chill Study Beat
GitHub:buttons
Choose a button
Follow
Watch
Follow
Watch
1J2GxbWa2o (1).gif
1J2GxbWa2o.gif
ss6fW8XoSL (1).gif

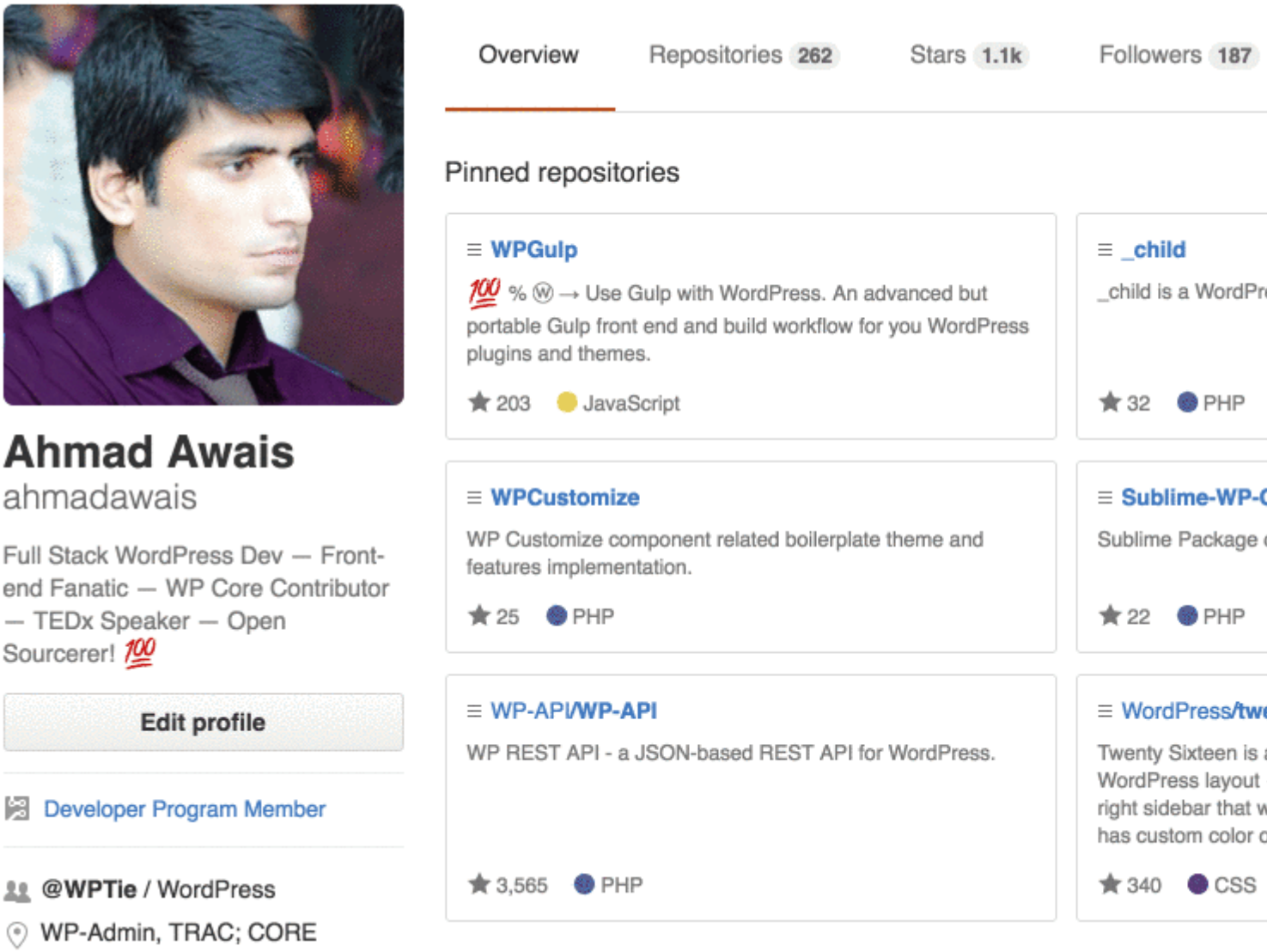
использование-кнопок-github

# глава 7: Как создать пользовательские ярлыки GitHub?

## Examples

### Создавайте пользовательские ярлыки GitHub!

Вот быстрый GIF, чтобы сделать процесс настолько простым, насколько это возможно.




Ярлыки могут применяться к проблемам и вызывать запросы, чтобы обозначать приоритет, категорию или любую другую информацию, которую вы считаете полезной.

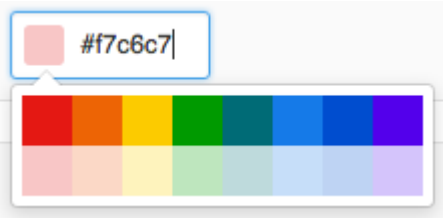
На GitHub перейдите на главную страницу репозитория.

1. Под своим именем репозитория нажмите «Проблемы» или «Запросы Pull».
2. Кнопка «Ярлыки проблем». Далее в поле поиска нажмите «Ярлыки».

3. Нажмите «Создать ярлык», чтобы создать новую метку, или нажмите «Изменить», чтобы изменить существующую.

4. В текстовом поле введите новое имя ярлыка.

5. Выберите цвет для метки из цветовой полосы. Вы можете настроить этот цвет, отредактировав шестнадцатеричное число над цветовой полосой.




6. Нажмите «Создать метку», чтобы сохранить новую метку.

Я надеюсь, что это помогает. Повысьте это, если это произойдет.

Прочитайте Как создать пользовательские ярлыки GitHub? онлайн:
https://riptutorial.com/ru/github/topic/7159/как-создать-пользовательские-ярлыки-github-

# глава 8: Клонирование хранилища от GitHub

## Синтаксис

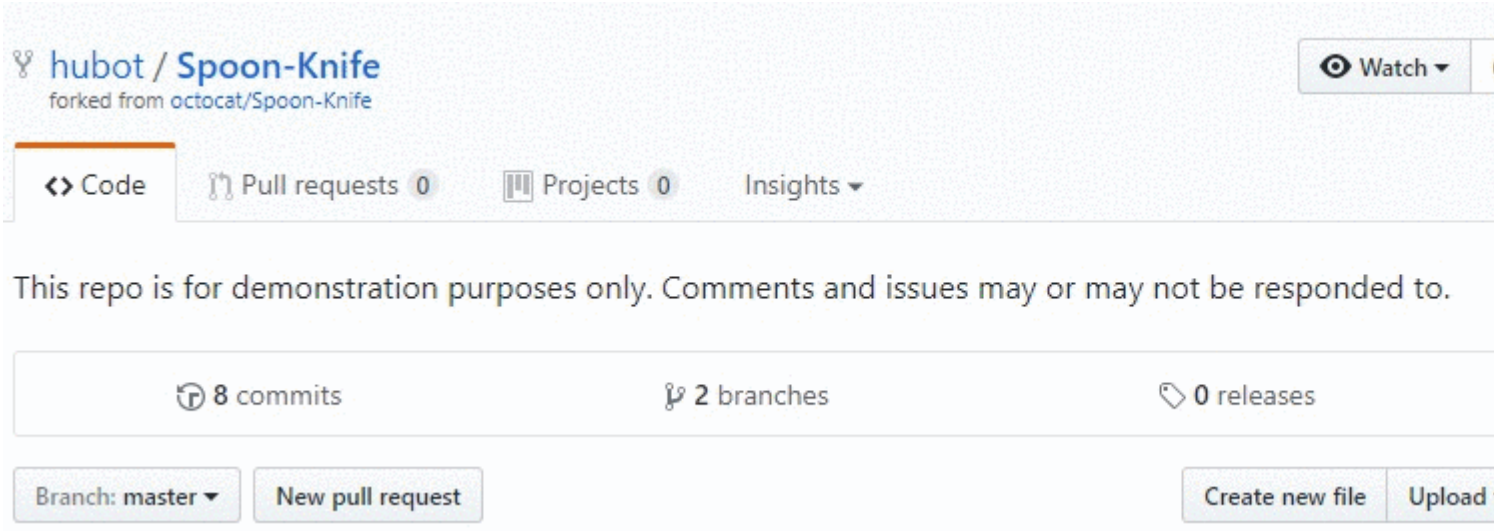
- git clone github.com/username/repository

## Examples

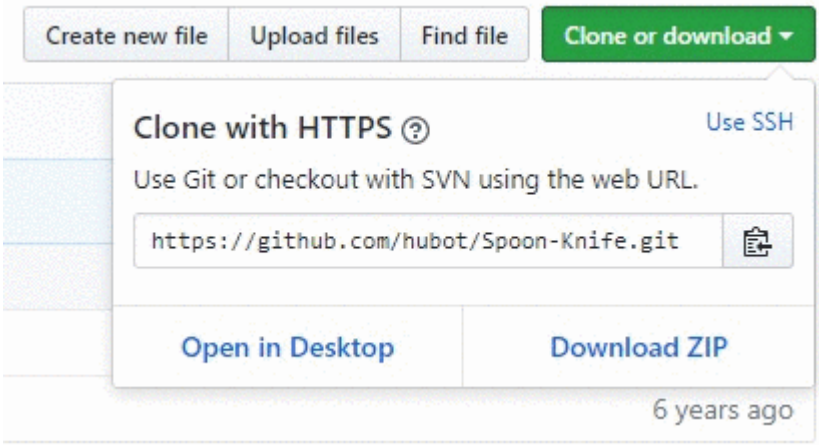
### Клонировать хранилище

1. Перейдите в репозиторий, который вы хотите клонировать (что-то вроде: https://github.com/ *username* / *repo* )


2. Справа нажмите зеленую кнопку с именем *clone или загрузите*




3. Появится небольшое окно, скопируйте URL-адрес (например: https://github.com/ *username* / *repo* .git)




4. Откройте окно терминала на компьютере, к которому вы хотите клонировать этот проект, чтобы
5. Перейдите из командной строки в место, к которому вы хотите клонировать проект.
6. Введите команду: git clone <copied_url_from_step_3>
7. нажмите Ввод
8. Появится следующее сообщение:

   Клонирование в `<repo_name>` ...

   remote: подсчет объектов: 10, сделано.

   remote: Сжатие объектов: 100% (8/8), сделано.

   удалить: Всего 10 (дельта 1), повторно использовать 10 (дельта 1)

   Распаковка объектов: 100% (10/10), сделано.

Прочитайте Клонирование хранилища от GitHub онлайн:
https://riptutorial.com/ru/github/topic/3761/клонирование-хранилища-от-github

# глава 9: Обновить разветвленный репозиторий

## замечания

- Справка GitHub: настройка пульта дистанционного управления для вилки
- Справка GitHub: синхронизация вилки
- популярные анны в StackOverFlow

## Examples

### Конфигурируйте пульт дистанционного управления для вилки, затем синхронизируйте свою вилку (мастер-ветвь)

1. Конфигурировать пульт для моей вилки

```
$ cd my_local_repo

$ git remote add upstream https://github.com/ORIGINAL_OWNER/ORIGINAL_REPOSITORY.git
    # Specify a new remote upstream repository that will be synced with the fork

$ git remote -v
    # Verify the new upstream repository specified for my fork
```

2. Синхронизировать свою вилку локально

```
$ cd my_local_repo

$ git fetch upstream
    # Fetch the branches and their respective commits from the upstream repository
    # Commits to master will be stored in a local branch, upstream/master

$ git checkout master

$ git merge upstream/master
    # Merge the changes from upstream/master into your local master branch
    # brings your fork's master branch into sync with the upstream repo
```

3. Синхронизировать вилку на Github

```
$ git push origin master
```

Прочитайте Обновить разветвленный репозиторий онлайн:
https://riptutorial.com/ru/github/topic/3758/обновить-разветвленный-репозиторий

# глава 10: Отображение временной шкалы / каналов GitHub на вашем веб-сайте

## Examples

### Отображение временной шкалы / каналов GitHub на вашем веб-сайте

В этом документе объясняется, как отображать ваши фиды / временные рамки GitHub на вашем веб-сайте.

**Пример:** живой пример доступен по адресу:

> https://newtonjoshua.com

**График GitHub:**

GitHub предоставляет публичную шкалу времени для любого пользователя в формате Atom.

Вы можете просмотреть свою временную шкалу:

> https://github.com/ {{GitHub_username}}. atom

Refer: https://developer.github.com/v3/activity/feeds

**API Google Feed:**

С API-интерфейсом Feed вы можете загружать любые публичные каналы Atom, RSS или Media RSS только с использованием JavaScript, поэтому вы можете размалывать фиды с вашим контентом и другими API-интерфейсами всего несколькими строками JavaScript. Это упрощает интеграцию фидов на вашем веб-сайте.

см. https://developers.google.com/feed/v1/devguide

**Загрузка JavaScript API:** для начала использования API фида включите следующий сценарий в заголовок вашей веб-страницы.

```
<script type="text/javascript" src="https://www.google.com/jsapi"></script>
```

Затем загрузите API фида с помощью google.load (модуль, версия, пакет).

```
<script type="text/javascript">
  google.load("feeds", "1");
</script>
```

**Указание URL-адреса фида:** вы можете вызвать google.feeds.Feed () следующим образом:

```
var feed = new google.feeds.Feed("https://github.com/{{GitHub_UserName}}.atom");
```

**Загрузка фида:** .load (обратный вызов) загружает фид, указанный в конструкторе с серверов Google, и вызывает данный обратный вызов, когда загрузка завершается.

```
<script type="text/javascript">

    function initialize() {
      feed.load(function(result) {
        if (!result.error) {
          var container = document.getElementById("feed");
          result.feed.entries.forEach(function (feed) {
            var feedTitle= feed.title;
            var feedLink = feed.link;
            var feedDate = formatDate(feed.publishedDate);
            var feedContent = formatContent(feed.content);

           // display the feed in your website
          });
        }
      });
    }
    google.setOnLoadCallback(initialize);

    </script>
```

**Вызов обработчика onLoad:** setOnLoadCallback (callback) - статическая функция, которая регистрирует указанную функцию обработчика, которая будет вызываться после того, как страница, содержащая эти нагрузки вызова, где обратный вызов является требуемой функцией, вызываемой при загрузке содержащего документа, и API готов к использованию

```
<script type="text/javascript">
    google.setOnLoadCallback(initialize);
 </script>
```

**Установка количества записей фида:** .setNumEntries (num) задает количество загружаемых этим фидом записей в num. По умолчанию класс Feed загружает четыре записи.

```
var feed = new google.feeds.Feed("https://github.com/{{GitHub_UserName}}.atom");
feed.setNumEntries(500);
```

Теперь вы можете форматировать и отображать свои фиды / временные рамки GitHub на своем веб-сайте.

Прочитайте Отображение временной шкалы / каналов GitHub на вашем веб-сайте онлайн: https://riptutorial.com/ru/github/topic/7479/отображение-временной-шкалы---каналов-github-на-вашем-веб-сайте

# глава 11: Работа с Gitflow

## Синтаксис

- git flow <подкоманда>
- git flow init
- git flow [feature | release | исправление] [начало | конец]

## параметры

| Подкоманда | подробности |
|---|---|
| в этом | Инициализация нового git-репо с поддержкой модели ветвления. |
| особенность | Управляйте филиалами функций. |
| релиз | Управляйте ветвями выпуска. |
| исправление | Управляйте ветвями исправлений. |

## замечания

- Концепция gitflow от автора
- изображение модели филиала

## Examples

### Операция на 5 общих ветвях локально

Один из наиболее распространенных случаев использования Gitflow

1. *Инициализировать* репо и определить ветви

```
$ git flow init
    # if you use default setup, you'll define six types of branches:
    #
    # main branches (lives forever)
    #
    #   1. master:  for production releases
    #   2. develop: for "next release" development
    #
    # supporting branches
    #
    #   3. feature: for a product feature
    #   4. release: for preparation of a new production release
```

```
    #   5. hotfix:  for resolving critical bug of production version
    #   6. support
    #
    # also, two main branches are created: master, develop
```

2. *Запуск* и *завершение* функции

```
$ git flow feature start my_feature
    # create branch 'feature/my_feature' based on the 'develop'

    # made development and commits...

$ git flow feature finish my_feature
    # merge 'feature/my_feature' back to the 'develop'
    # delete 'feature/my_feature'
```

3. *Запустить* и *закончить* выпуск

```
$ git flow release start my_release
    # create branch 'release/my_release' based on the 'develop'

    # made bug fixes...

$ git flow release finish my_release
    # merge branch 'release/my_release' to the 'master' and add tag
    # merge branch 'release/my_release' back to the 'develop'
    # delete 'release/my_release'
```

4. *Запустить* и *завершить* исправление

```
$ git flow hotfix start my_hotfix
    # create branch 'hotfix/my_hotfix' based on the 'master'

    # made some hotfixes...

$ git flow hotfix finish my_hotfix
    # merge branch 'hotfix/my_hotfix' back to the 'master' and add tag
    # merge branch 'hotfix/my_hotfix' to the 'develop'
    # delete 'hotfix/my_hotfix'
```

Прочитайте Работа с Gitflow онлайн: https://riptutorial.com/ru/github/topic/6231/работа-с-gitflow

# глава 12: Рабочий стол GitHub

## Вступление

Как установить и работать с GitHub Desktop?

Рабочий стол GitHub - это название - среда рабочего стола для Windows и MacOS, которая включает в себя основные функции Git, такие как клонирование, нажатие, вытягивание (синхронизация в рабочем столе GitHub), слияние ...

Основная цель клиентов Desktop - предоставить более простой способ работы с git (и GitHub). В фоновом режиме он использует те же команды, что и большинство пользователей, которые будут использовать из командной строки.

## Examples

### Установка и настройка

Установка довольно проста , как Есть отдельные инсталляторы для MacOS и Windows , машин , доступных здесь . В настоящее время для скачивания доступны две версии: одна бета и одна стабильная.

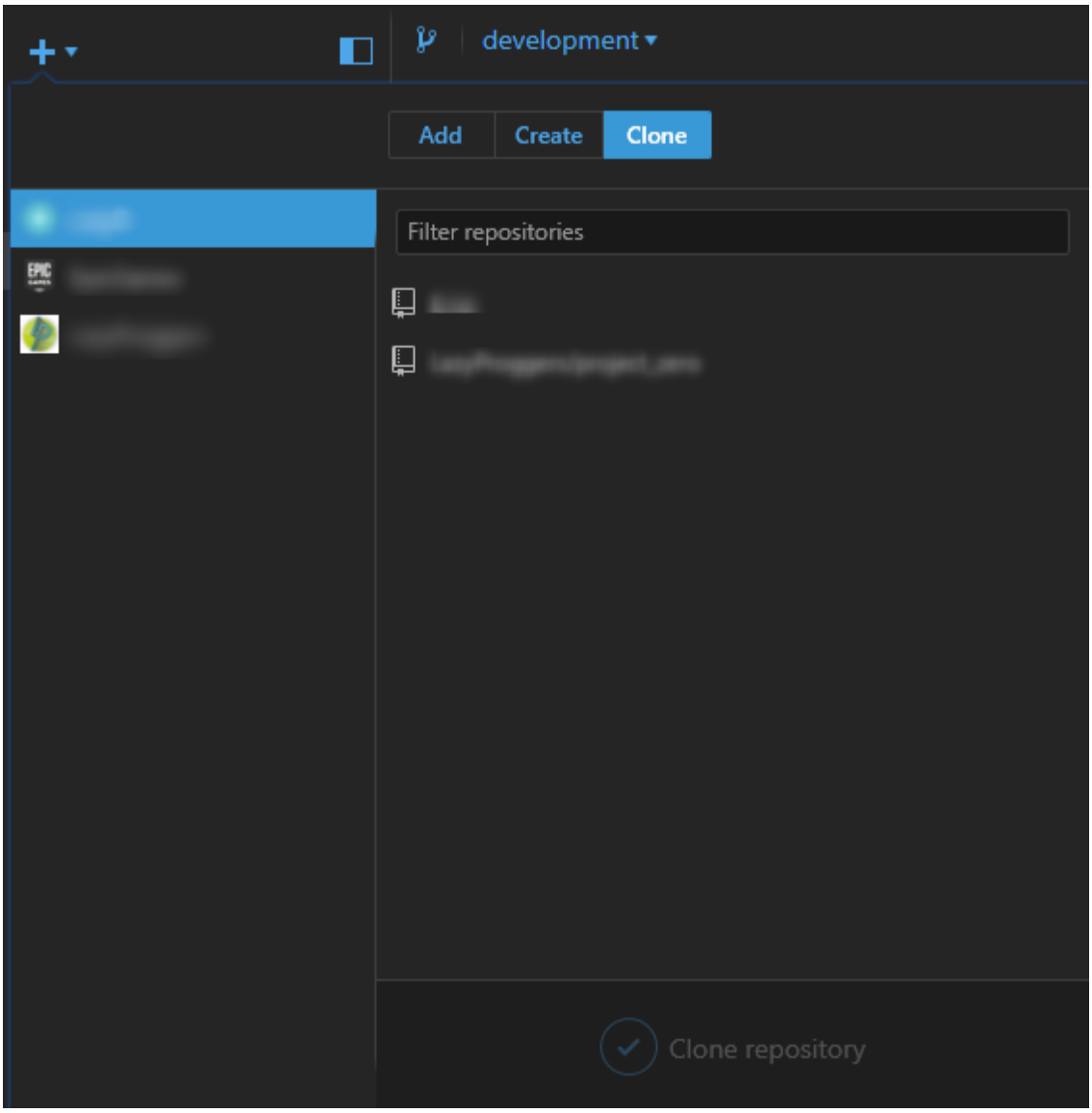
Установка начнется после загрузки программы, и вам нужно будет войти в систему со своими учетными данными GitHub. Это действительно единственный шаг, потому что после этого вы можете начать создавать репозиторий или клонировать его.

> Примечание: во время установки будет установлен не только GitHub Desktop, но и Git. Поэтому вам не нужно устанавливать его отдельно.

### Клонирование хранилища

Как и в случае с GitHub Desktop, большая часть работы довольно проста: вы выбираете « Clone a repository» (в стабильной версии плюс в левом верхнем углу), и есть некоторые репозитории (ваши собственные и репозитории от каждой компании, в которой вы находитесь ) рекомендуемые. Кроме того, вы можете вставить ссылку на любой другой репозиторий, который вы захотите клонировать.

> Примечание: в более новой версии (бета) нет рекомендаций (не jet?).




## разветвление

Вы можете выбрать ветку в левом верхнем углу. Когда вы выбрали нужную ветку, вам нужно нажать кнопку синхронизации (в правом верхнем углу), которая теперь совпадает с `git checkout BRANCHNAME` .

> В более старой версии вы можете одновременно просмотреть две разные ветви и сравнить толки. Кроме того, вы можете просмотреть график своего проекта (см. Ниже)


### Создание новой ветки

Вы можете создать новую ветку, нажав на символ ветки (старый клиент) или в разделе « `File --> New Branch` .

> Обратите внимание, что вы можете выбрать ветку, которую новая ветка использует в качестве базы, нажав на название ветки.

## Push и Pull (или: кнопка синхронизации)

### Pull (Sync)

Как и в командной строке, вам нужно вытягивать текущее состояние репозитория один раз в то время. В GitHub Desktop этот процесс вызывается кнопкой `sync` в верхнем правом углу.

### От себя

Когда вы делаете локальные изменения и хотите их нажимать, вы делаете фиксацию, записывая что-то в сводном текстовом поле. Затем вы нажимаете `Commit to YOURCURRENTBRANCH` . Теперь вам нужно нажать кнопку синхронизации и сделать нажатие.

> Примечание. Вы можете использовать смайлики, упоминания и рефералы для других коммитов или проблем непосредственно из текстового поля.

Таким образом, кнопка **Sync** может использоваться для `Push` , `Pull` или `Checkout` .

Прочитайте Рабочий стол GitHub онлайн: https://riptutorial.com/ru/github/topic/10023/рабочий-стол-github

# глава 13: Резервное копирование GitHub

## Examples

### Клонирование всех репозиториев для имени пользователя

Выполните следующую команду, заменив имя пользователя на имя пользователя, чтобы клонировать все репозитории GitHub для этого пользователя в текущий каталог.

```
curl "https://api.github.com/users/username/repos?page=1&per_page=100" | grep -e 'git_url*' |
cut -d \" -f 4 | xargs -L1 git clone
```

Это приведет к клонированию первых 100 репозиториев.

Прочитайте Резервное копирование GitHub онлайн: https://riptutorial.com/ru/github/topic/3760/резервное-копирование-github

# глава 14: Страницы GitHub

## Examples

### Использование автоматического генератора страниц для хранилища

1. Перейти на сайт GitHub
2. Откройте свой репозиторий
3. Нажмите «Настройки»
4. На странице GitHub нажмите «Запустить автоматический генератор страниц»,
5. Следуй инструкциям

### Использование Git для создания страниц с нуля

1. Создайте новый репозиторий или клонируйте существующий.
2. Создайте новую ветвь, называемую `gh-pages` без какой-либо истории

```
$ git checkout --orphan gh-pages

# ensure you are in the correct directory then,
# remove all files from the old working tree
$ git rm -rf
```

3. Добавьте файл index.html в корень репозитория.

```
$ echo "Hello World" > index.html
$ git add index.html
$ git commit -a -m "First pages commit"
```

4. Нажмите на Github.

```
$ git push origin gh-pages
```

Теперь вы можете загрузить новый сайт Github Pages по адресу
`http(s)://<username>.github.io/<projectname>`

### Создание настраиваемого URL-адреса для вашей страницы GitHub

Вам потребуется доменное имя от регистратора .

В ветке `gh-pages` вашего репозитория проекта или главной ветви вашего хранилища `username.github.io` создайте файл CNAME с содержимым `www.yourdomain.com` - *канонический домен* .

На странице конфигурации домена вашего регистратора укажите свой домен на свой веб-

сайт GitHub. Настройте две записи CNAME (один для корневой вершины (@) и один для www). Оба должны указывать на `username.github.io` или `username.github.io/repository` . Если поставщик DNS НЕ поддерживает записи ALIAS на вершине корня (@), просто создайте записи A, которые указывают на 192.30.252.153 и 192.30.252.154.

---

## Ресурсы

Инструкции GitHub для пользовательского домена

Переполнение стека Q & A: «Пользовательский домен для страниц проекта GitHub»

Одри Уоттерс - Использование GitHub для создания веб-проекта: как и почему

Alex Cican - Как я переместил свои сайты в Dropbox и GitHub

Treehouse - Использование страниц GitHub для размещения вашего сайта

Прочитайте Страницы GitHub онлайн: https://riptutorial.com/ru/github/topic/3759/страницы-github

# глава 15: Удаление конфиденциальных данных или больших файлов

## Вступление

Если вы отправляете важные данные, такие как пароль или SSH-ключ, в репозиторий Git, вы можете удалить его из истории. Чтобы полностью удалить нежелательные файлы из истории репозитория, вы можете использовать команду git filter-branch или BFG Repo-Cleaner.

## замечания

1. Сообщите своим сотрудникам об отказе, а не объединении, каких-либо ветвей, которые они создали из старой (испорченной) истории хранилища. Одно объединение слияния может повторно ввести какую-то или всю испорченную историю, в которую вы только что столкнулись с проблемой очистки.

2. По прошествии некоторого времени вы уверены, что git filter-branch не имеет непредвиденных побочных эффектов, вы можете заставить все объекты в вашем локальном репозитории разыменоваться и собирать мусор с помощью следующих команд (с использованием Git 1.8.5 или новее):

   git for-each-ref --format = 'delete% (refname)' refs / original | git update-ref --stdin

   git reflog expire --expire = now - all

   git gc --prune = сейчас

## Examples

### Использование фильтра-ветви

```
git filter-branch --force --index-filter \
'git rm --cached --ignore-unmatch PATH-TO-YOUR-FILE-WITH-SENSITIVE-DATA' \
--prune-empty --tag-name-filter cat -- --all
```

Добавьте свой файл с конфиденциальными данными в .gitignore, чтобы убедиться, что вы его случайно не совершили.

```
echo "YOUR-FILE-WITH-SENSITIVE-DATA" >> .gitignore
git add .gitignore
git commit -m "Add YOUR-FILE-WITH-SENSITIVE-DATA to .gitignore"
```

Нажмите свое местное репо на GitHub

```
git push origin --force --all
```

Чтобы удалить чувствительный файл из ваших отмеченных выпусков, вам также потребуется принудительно нажать на теги Git:

```
git push origin --force --tags
```

### Использование BFG Repo Cleaner

Очиститель BFG Repo является альтернативой git filter-branch. Его можно использовать для удаления конфиденциальных данных или больших файлов, которые были сделаны неправильно, как двоичные файлы, скомпилированные из источника. Это написано в Scala.

Веб-сайт проекта: BFG Repo Cleaner

## Требования

Java Runtime Environment (Java 7 или выше - BFG v1.12.3 была последней версией для поддержки Java 6). Библиотека Scala и все другие зависимости складываются в загружаемую банку.

## Удаление файлов с конфиденциальными данными

```
bfg --delete-files YOUR-FILE-WITH-SENSITIVE-DATA
```

Прочитайте Удаление конфиденциальных данных или больших файлов онлайн: https://riptutorial.com/ru/github/topic/8170/удаление-конфиденциальных-данных-или-больших-файлов

# кредиты

| S. No | Главы | Contributors |
|---|---|---|
| 1 | Начало работы с github | BadAllOff, BrechtDeMan, Community, H. Pauwelyn, Hamzawey, Hugo, intboolstring, Kronos, Mateusz Piotrowski, Minhas Kamal, Nicholas Qiao, rpadovani |
| 2 | вопросы | geek1011, Hamzawey, SuperBiasedMan |
| 3 | Вытянуть запросы | Maxcell |
| 4 | загрузить один файл из репозитория GitHub | ownsourcing dev training |
| 5 | Использование Gist | Kronos, tehp |
| 6 | Использование кнопок GitHub | James Kerrane |
| 7 | Как создать пользовательские ярлыки GitHub? | Ahmad Awais |
| 8 | Клонирование хранилища от GitHub | demonplus, geek1011, Hamzawey, James Kerrane, Mateusz Piotrowski |
| 9 | Обновить разветвленный репозиторий | Derek Liu |
| 10 | Отображение временной шкалы / каналов GitHub на вашем веб-сайте | Hugo, Newton Joshua |
| 11 | Работа с Gitflow | Derek Liu |
| 12 | Рабочий стол GitHub | creyD |

| | | |
|---|---|---|
| 13 | Резервное копирование GitHub | geek1011 |
| 14 | Страницы GitHub | BrechtDeMan, geek1011, Mono |
| 15 | Удаление конфиденциальных данных или больших файлов | Gautam Krishna R, Kronos |