**FREE eBook**

# LEARNING

# github

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

#github

# Table of Contents


About ... 1

Chapter 1: Getting started with github ... 2

Remarks ... 2

Examples ... 2

Installation or Setup ... 2

Creating An Account ... 2

Useful Tools ... 2

Creating Your First Repository ... 2

Online ... 2

Offline ... 2

README file ... 3

A README file may include- ... 3

Project Title ... 3

Download ... 3

Installation ... 3

Demonstration ... 3

Authors ... 3

Acknowledgments ... 4

Contributing ... 4

License ... 4

LICENSE file ... 4

GitHub Flavored Markdown ... 5

Header ... 5

Emphasis ... 6

Horizontal Line ... 6

List ... 7

Table ... 8

Code ... 8

Quote ... 9

Link ... 9
Image ... 9
Task Lists ... 10
Emoji ... 10
SHA references ... 10
Pull Request and Issue References ... 10
Chapter 2: Backing up GitHub ... 12
Examples ... 12
Cloning all repositories for a username ... 12
Chapter 3: Cloning a repository from GitHub ... 13
Syntax ... 13
Examples ... 13
Clone a repository ... 13
Chapter 4: Displaying GitHub timeline / feeds in your Website ... 15
Examples ... 15
Displaying GitHub timeline / feeds on your website ... 15
Chapter 5: download single file from GitHub repository ... 17
Examples ... 17
from a public repository using the command line and renaming file ... 17
find the url of the file you want to download ... 17
Chapter 6: GitHub Desktop ... 18
Introduction ... 18
Examples ... 18
Installation and Setup ... 18
Cloning a repository ... 18
Branching ... 19
Push and Pull (or: the Sync Button) ... 20
Chapter 7: GitHub Pages ... 21
Examples ... 21
Using the automatic page generator for a repository ... 21
Using Git to create pages from scratch ... 21

Creating a custom URL for your GitHub page ... 21
Resources ... 22
Chapter 8: How to create custom GitHub Labels? ... 23
Examples ... 23
Create Custom GitHub Labels! ... 23
Chapter 9: Issues ... 25
Examples ... 25
Creating an issue ... 25
Chapter 10: Pull Requests ... 26
Examples ... 26
Opening a Pull Request ... 26
New Pull Request ... 26
h21 ... 27
Chapter 11: Removing sensitive data or large files ... 29
Introduction ... 29
Remarks ... 29
Examples ... 29
Using filter-branch ... 29
Using the BFG Repo Cleaner ... 30
Requirements ... 30
Remove files with sensitive data ... 30
Chapter 12: Update a forked Repository ... 31
Remarks ... 31
Examples ... 31
Config a remote for your fork then sync your fork (master branch) ... 31
Chapter 13: Using Gist ... 32
Introduction ... 32
Remarks ... 32
Examples ... 32
Public Gist ... 32
Secret Gist ... 33

Chapter 14: Using GitHub Buttons ..... 34
Introduction ..... 34
Remarks ..... 34
Examples ..... 34
Follow Button ..... 34
All Other Buttons ..... 36
Chapter 15: Working with Gitflow ..... 39
Syntax ..... 39
Parameters ..... 39
Remarks ..... 39
Examples ..... 39
Operation on 5 common branches locally ..... 39
Credits ..... 41

# About

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

# Chapter 1: Getting started with github

## Remarks

This section provides an overview of what github is, and why a developer might want to use it.

It should also mention any large subjects within github, and link out to the related topics. Since the Documentation for github is new, you may need to create initial versions of those related topics.

## Examples

### Installation or Setup

GitHub is a huge collection of Git repositories. In other words, you can think of GitHub as a collection of many projects!

# Creating An Account

- Visit GitHub's main page Here
- Pick a username, enter in your email address, and pick a secure password and you're ready to go!

# Useful Tools

For Git/GitHub beginners, understanding how version control works might be confusing at first. There exists a GUI version of GitHub that you can download and use. GitHub Desktop is just that tool.

# Creating Your First Repository

You can think of a repository as a project. You can create a repository online or offline. Follow the steps below:

## Online

1. First log in and go to your profile.
2. Navigate to the "Repositories" tab near the top of the page
3. Hit the green "New" button and you're ready to rumble!

## Offline

1. Download and install git (choose the operating system you are running)
2. After downloading and installation, you can either use the command line tool, or you can download a GUI client.
3. After installation, create an account on github
4. From the top right, click on the + and choose either creating a new repository or import an existing on.
5. If you choose a new one, enter the repository name and choose either to have it public or private.
6. Click: Create Repository

N.B. Private repositories are not available for free users.

### README file

If your project doesn't have README.md, GitHub may parse README.rdoc to display details. If it has both, it will use README.md, silently ignoring rdoc.

---

# A README file may include-

## Project Title

Describe briefly about your project. You may also provide project's website link, badges, community & contact info (i.e. email, social site).

## Download

Runnable file (executable or minified or installation file) link. There can be links to previous versions too.

## Installation

How your work can be used. It may include the prerequisites, settings, third party libraries, usage, cautions, etc.

## Demonstration

It may include code sample, gif file, video link, or even screen shots.

## Authors

Author names, contact info, etc.

## Acknowledgments

List of people or community helped and inspired throughout the project

## Contributing

Instructions to contribute (i.e. add feature, report bug, submit patch) to the project. May include documentation link too.

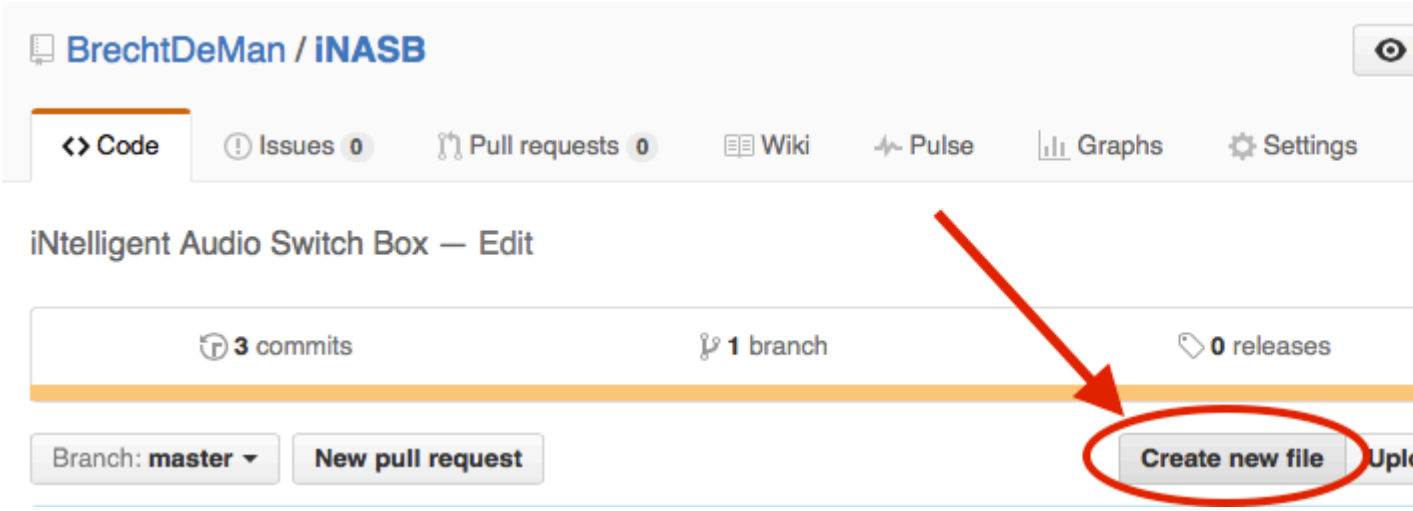
## License

Give a short intro over your license. You can give a link to the license site too.

### LICENSE file

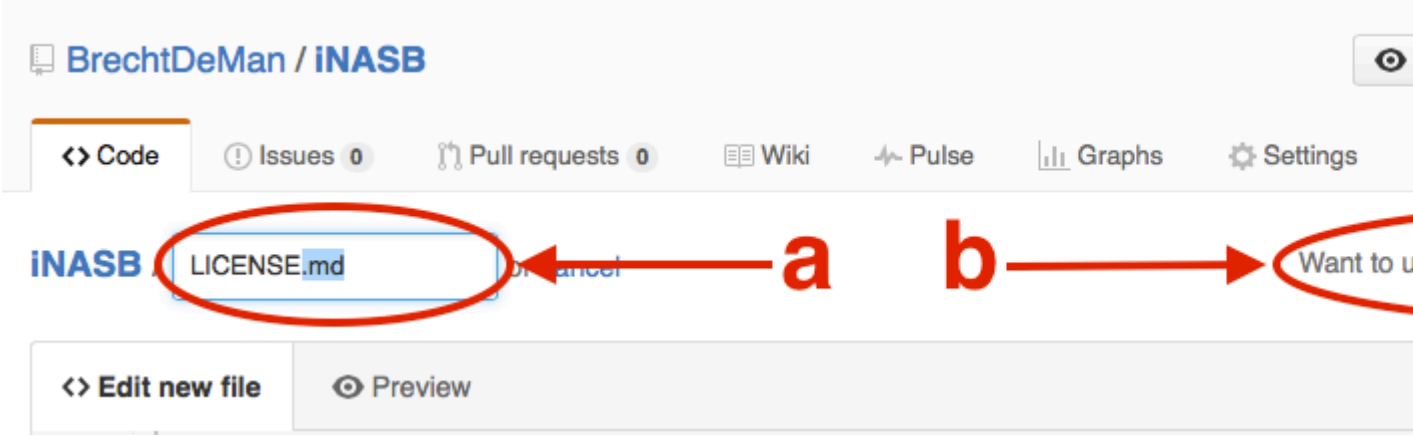
GitHub helps you quickly add a license to your repository, as an alternative for adding your own text/markdown file.

1. In your repository, click 'Create new file'

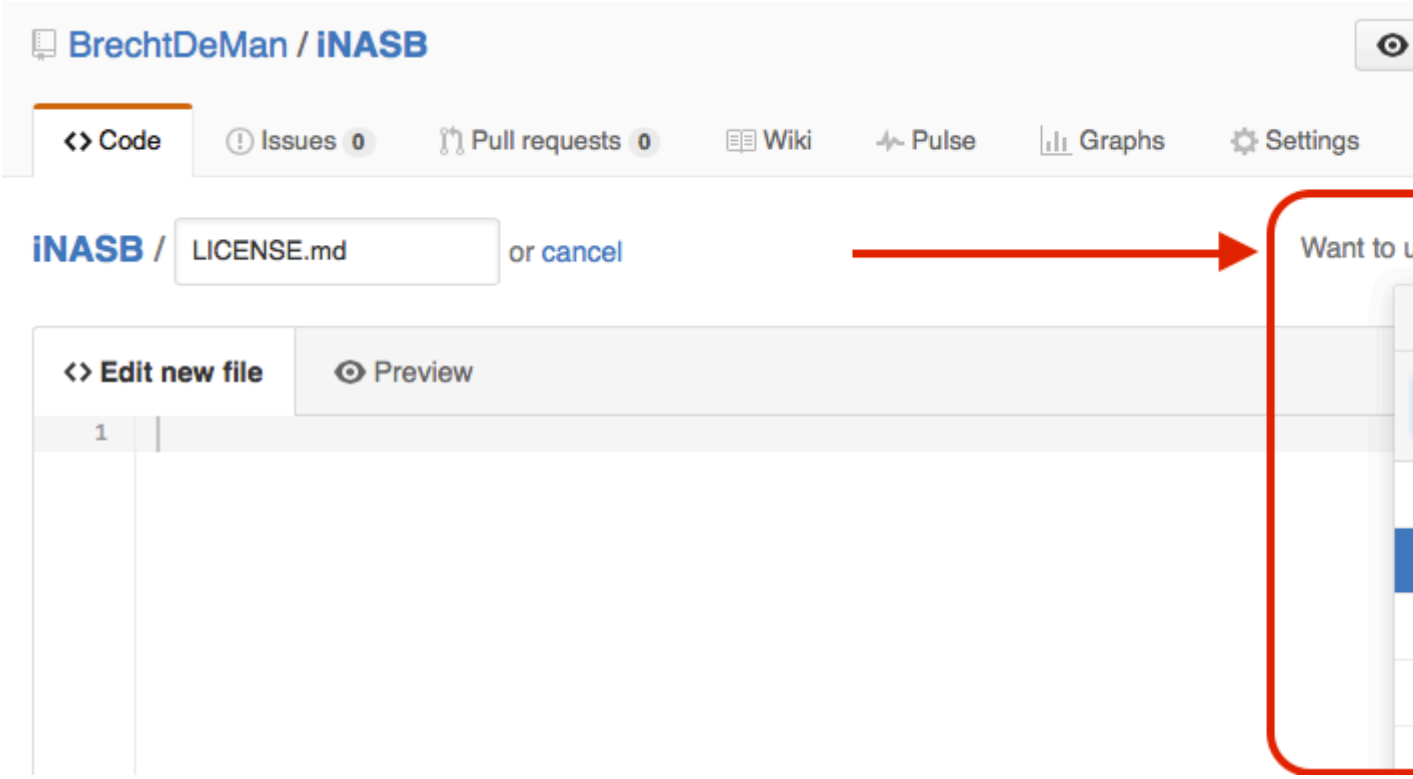



2. On next page:

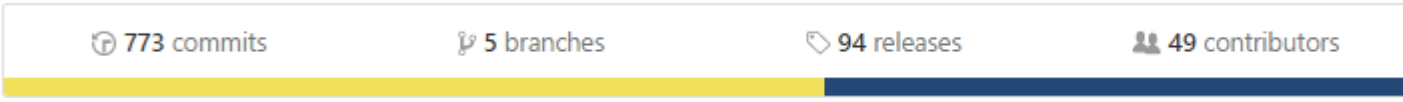
    1. Type `LICENSE.md` or `LICENSE.txt` as the new file's file name.
    2. The *Want to use a new template?* dialog will appear.




3. Choose your preferred license.




4. The licence you could see in the repository details:




*From Q&A - How to add license to a existing Github project*

### GitHub Flavored Markdown

GitHub expands Markdown syntax to provide new useful features.

# Header

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

# Header1

## Header2

### Header3

#### Header4

##### Header5

###### Header6

# H1

## H2

# Emphasis

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

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

# Horizontal Line

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

---

---

---

# List

```
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

# Table

```
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 |

# Code

```
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");
}
```

# Quote

```
> 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.

# Link

```
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

# Image

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

# Task Lists

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

- [x] completed item
- [ ] incomplete item

# Emoji

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

For all GitHub emojies visit- Emoji Cheat Sheet.

# SHA references

Any reference to a SHA1 hash of a commit will be converted into a link to the commit itself on GitHub:

```
e7909ea4fbb162db3f7f543d43c30684a3fb745f
```

Write Preview

`e7909ea`

# Pull Request and Issue References

Any reference to a pull request or an issue will automatically be linked to that pull request or issue.

This can be done by putting a `#` in front of the issue/Pull Request number.

Read Getting started with github online: https://riptutorial.com/github/topic/1214/getting-started-with-github

# Chapter 2: Backing up GitHub

## Examples

### Cloning all repositories for a username

Run the following command, replacing username with the username, to clone all of the GitHub repositories for that user to the current directory.

```
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
```

This will only clone the first 100 repositories.

Read Backing up GitHub online: https://riptutorial.com/github/topic/3760/backing-up-github

# Chapter 3: Cloning a repository from GitHub

## Syntax

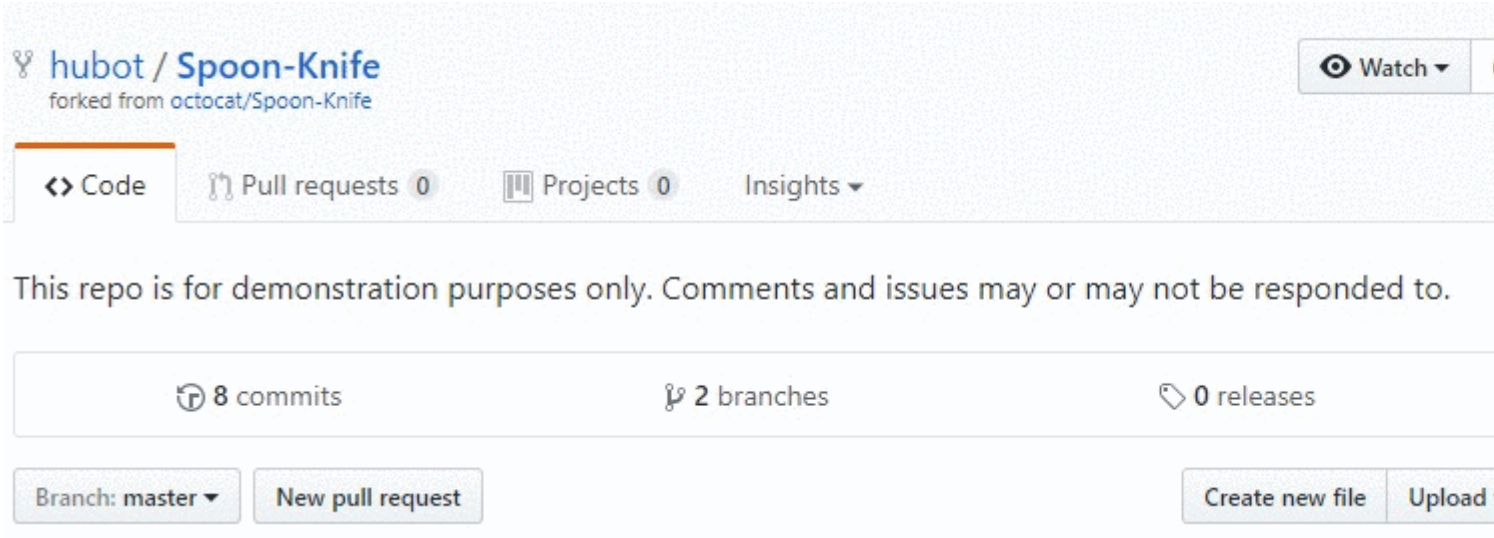
- git clone github.com/username/repository

## Examples

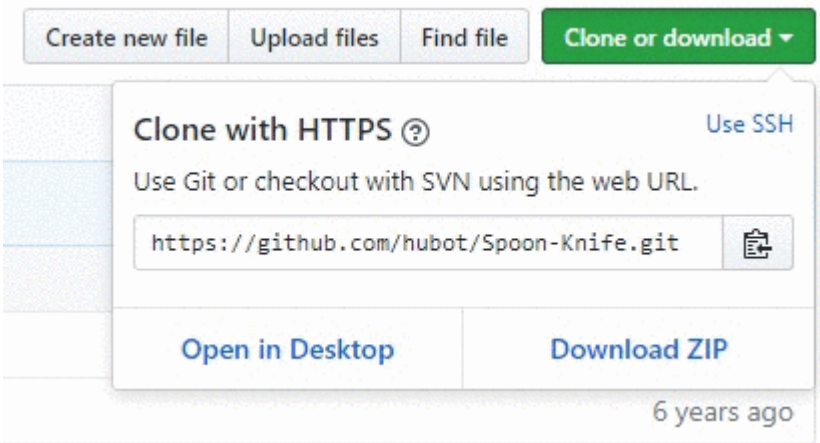
### Clone a repository

1. Go to the repository you want to clone (something like: https://github.com/*username*/*repo*)


2. On the right, click on the green button named *clone or download*




3. A small window will appear, copy the url (something like: https://github.com/*username*/*repo*.git)



4. Open a terminal window on the machine you want to clone that project to
5. Navigate from the command line to the location you want to clone the project to
6. Enter the command: git clone <copied_url_from_step_3>
7. Press Enter
8. Something like the following will appear:

   Cloning into `<repo_name>`...

   remote: Counting objects: 10, done.

   remote: Compressing objects: 100% (8/8), done.

   remove: Total 10 (delta 1), reused 10 (delta 1)

   Unpacking objects: 100% (10/10), done.

Read Cloning a repository from GitHub online: https://riptutorial.com/github/topic/3761/cloning-a-repository-from-github

# Chapter 4: Displaying GitHub timeline / feeds in your Website

## Examples

### Displaying GitHub timeline / feeds on your website

This document explains how to display your GitHub feeds/timeline on your website.

**Example:** A live example is available at:

> https://newtonjoshua.com

**GitHub timeline:**

GitHub provides the public timeline for any user in Atom format.

You can view your timeline at:

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

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

**Google Feed API:**

With the Feed API, you can download any public Atom, RSS, or Media RSS feed using only JavaScript, so you can mash up feeds with your content and other APIs with just a few lines of JavaScript. This makes it easy to quickly integrate feeds on your website.

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

**Loading the JavaScript API:** To begin using the Feed API, include the following script in the header of your web page.

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

Next, load the Feed API with google.load(module, version, package).

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

**Specifying the feed URL:** You can call google.feeds.Feed() as follows:

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

**Loading a feed:** .load(callback) downloads the feed specified in the constructor from Google's

servers and calls the given callback when the download completes.

```
<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>
```

**Calling the onLoad handler:** setOnLoadCallback(callback) is a static function that registers the specified handler function to be called once the page containing this call loads, where callback is a required function called when the containing document is loaded and the API is ready for use

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

**Setting the number of feed entries:** .setNumEntries(num) sets the number of feed entries loaded by this feed to num. By default, the Feed class loads four entries.

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

Now you can format and display your GitHub feeds/timeline on your website.

Read Displaying GitHub timeline / feeds in your Website online:
https://riptutorial.com/github/topic/7479/displaying-github-timeline---feeds-in-your-website

# Chapter 5: download single file from GitHub repository

## Examples

### from a public repository using the command line and renaming file

This example grabs the Node.gitignore file from GitHub's gitignore repository, downloads it to your current working directory and renames it to .gitignore - all very typical actions for someone starting a new node.js project.

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

### find the url of the file you want to download

1. navigate to the desired file in a repository
2. click the 'raw' button
3. copy the url from the address bar

See the following example from GitHub's gitignore repository:
http://github.com/github/gitignore/raw/master/Node.gitignore

You can quickly recognize a url that will work to download an individual file vs downloading the html page. Look for the subdirectory /raw/ right before the branch name.

Read download single file from GitHub repository online:
https://riptutorial.com/github/topic/10898/download-single-file-from-github-repository

# Chapter 6: GitHub Desktop

## Introduction

How to install and work with GitHub Desktop?

GitHub Desktop is -as the name implies- an desktop environment for Windows and MacOS which includes the main features of Git like cloning, pushing, pulling (sync in GitHub Desktop), merging...

The Desktop clients main purpose is to deliver an easier way of working with git (and GitHub). In the background it uses the same commands as most users would use from the commandline.

## Examples

### Installation and Setup

The installation is quite simple as there are seperate installers for MacOS and Windows machines available here. Currently two versions are for download: one beta and one stable.

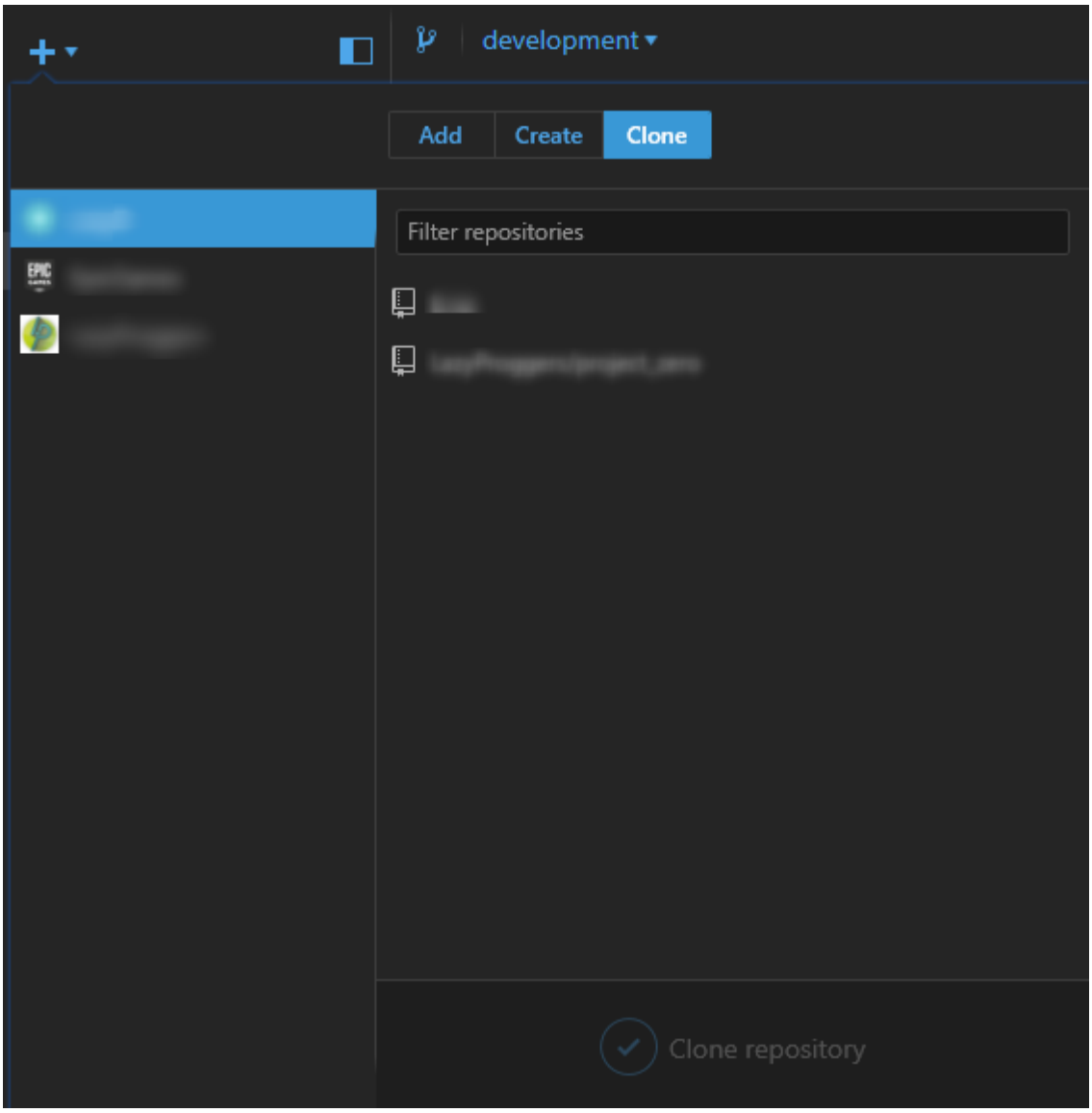
Setup will start after you downloaded the program and you´ll need to login with your GitHub credentials. That is really the only step because after that you can start to create a repository or clone one.

> Note: during the installation not only GitHub Desktop will be installed but Git too. So you don´t need to install it seperate.

### Cloning a repository

As it is with GitHub Desktop most of the work is pretty simple: You select "Clone a repository" (In the stable version the plus on the upper left) and there are some repositories (your own and the repos from each company you are in) recommended. Alternatively you can paste a link to any other repository you might want to clone.

> Note: in the newer version (beta) there are no (not jet?) recommendations.




## Branching

You can select a branch at the upper left. When you selected the right branch you need to press the sync button (upper right) which does now the same as `git checkout BRANCHNAME`.

> In the older version you are able to view 2 different branches at once and compare the pushes. Furthermore you could view a timeline of your project (see beneath)


### Creating a new branch

You can create a new branch by clicking on the branch symbol (old client) or under `File --> New Branch`.

> Note that you can select of which branch the new branch uses as base by clicking on the branch name.

## Push and Pull (or: the Sync Button)

### Pull (Sync)

Like in the command line you need to pull the current state of the repository once in a while. In GitHub Desktop this process is called by the `sync` Button at the top right corner.

### Push

When you made local changes and want to push them you make a commit by writing something into the summary textbox. Then you press `Commit to YOURCURRENTBRANCH`. Now you´ll need to press the sync button and your push is made.

> Note: You can use emoticons, mentions and referals to other commits or issues directly from the textbox.

So the **Sync** button can be used to `Push`,`Pull` or `Checkout`.

Read GitHub Desktop online: https://riptutorial.com/github/topic/10023/github-desktop

# Chapter 7: GitHub Pages

## Examples

### Using the automatic page generator for a repository

1. Go to the GitHub website
2. Open your repository
3. Click Settings
4. Under GitHub Pages, click "Launch Automatic Page Generator"
5. Follow the instructions

### Using Git to create pages from scratch

1. Create a new repository or clone an existing one.
2. Create a new branch called `gh-pages` without any history

```
$ 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. Add an index.html file to the root of the repository.

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

4. Push to Github.

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

You can now load your new Github Pages site at `http(s)://<username>.github.io/<projectname>`

### Creating a custom URL for your GitHub page

You will need a domain name from a registrar.

In the `gh-pages` branch of your project repository, or the main branch of your `username.github.io` repository, create a CNAME file with the contents `www.yourdomain.com` - the *canonical* domain.

At your registrar's domain configuration page, point your domain to your GitHub website. Set up two CNAME records (one for the root apex (@) and one for www). Both should point to `username.github.io` or `username.github.io/repository`. If your DNS provider does NOT support ALIAS records on the root apex (@), simply create A records that point to 192.30.252.153 and 192.30.252.154.

## Resources

GitHub instructions for a custom domain

Stack Overflow Q&A: "Custom domain for GitHub project pages"

Audrey Watters - Using GitHub to Power A Web Project: How and Why

Alex Cican - How I moved my websites to Dropbox and GitHub

Treehouse - Using GitHub Pages To Host Your Website

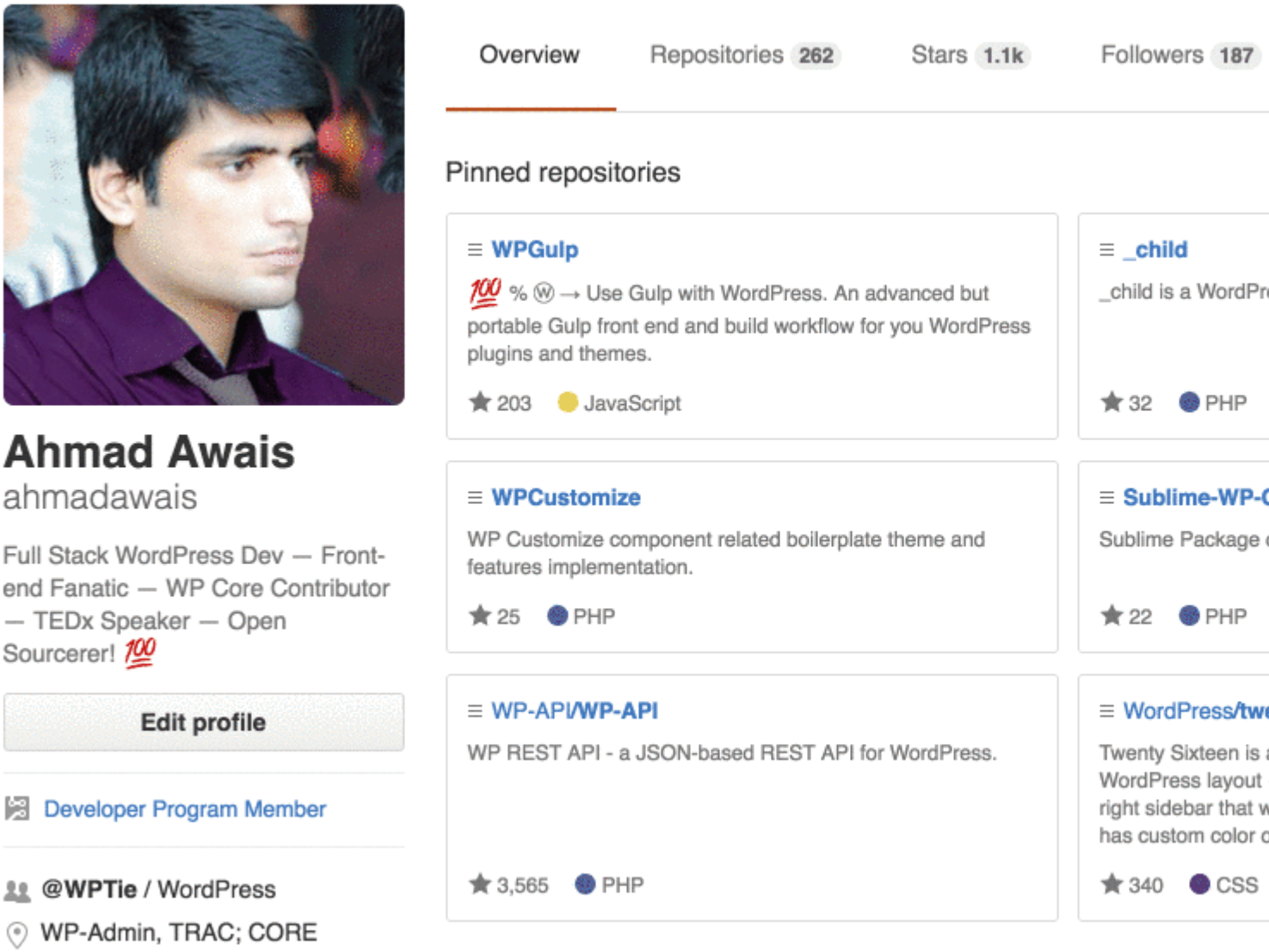
Read GitHub Pages online: https://riptutorial.com/github/topic/3759/github-pages

# Chapter 8: How to create custom GitHub Labels?

## Examples

### Create Custom GitHub Labels!

Here's a quick GIF to make the process as easy as it can be.

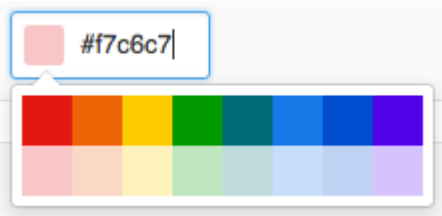



Labels can be applied to issues and pull requests to signify priority, category, or any other information that you find useful.

On GitHub, navigate to the main page of the repository.

1. Under your repository name, click Issues or Pull requests.
2. Issues Labels buttonNext to the search field, click Labels.

3. Click New Label to create a new label, or click Edit to edit an existing one.
4. In the text box, type your new label name.
5. Select a color for the label from the color bar. You can customize this color by editing the hexadecimal number above the color bar.




6. Click Create Label to save the new label.

I hope it helps. Upvote it if it does.

Read How to create custom GitHub Labels? online: https://riptutorial.com/github/topic/7159/how-to-create-custom-github-labels-

# Chapter 9: Issues

## Examples

### Creating an issue

1. Go to the GitHub page for the project where you want to create an issue.
2. Click on ***Issues*** .
3. On the top right, click on ***New Issue***.
4. Enter the title of the issue.
5. Enter the body of the issue (including logs, code snippets, etc.)
6. *Optional:* To view the issue before submitting it, click on preview.
7. Click on ***Submit new issue***.

Read Issues online: https://riptutorial.com/github/topic/3757/issues

# Chapter 10: Pull Requests

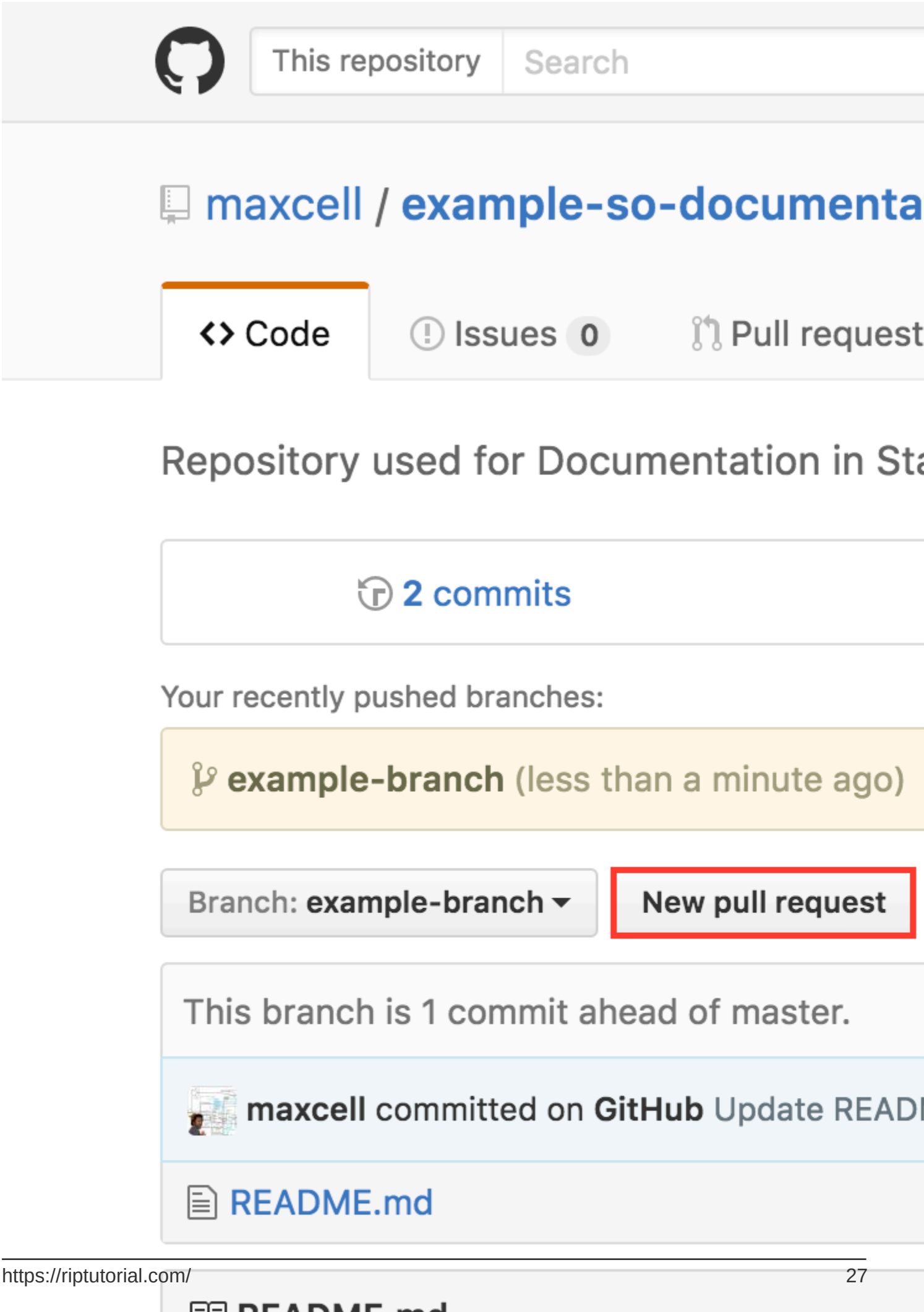
## Examples

### Opening a Pull Request

## New Pull Request

Whenever you want to create a Pull Request (let's say this is either by a recent change. But, you can also do this with an older change too!), you can go ahead and let GitHub do a lot of the heavy lifting for you and hit the green **Compare & Pull Request** button (*NOT TO BE CONFUSED WITH CLONE OR DOWNLOAD*) within the alert box that mentions that you just pushed within a branch.

Otherwise, you will use the **New Pull Request** button found next to your branch.


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

Update README.md

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting

Styling with Markdown is supported

Read Pull Requests online: https://riptutorial.com/github/topic/5761/pull-requests

# Chapter 11: Removing sensitive data or large files

## Introduction

If you commit sensitive data, such as a password or SSH key into a Git repository, you can remove it from the history. To entirely remove unwanted files from a repository's history you can use either the git filter-branch command or the BFG Repo-Cleaner.

## Remarks

1. Tell your collaborators to rebase, not merge, any branches they created off of your old (tainted) repository history. One merge commit could reintroduce some or all of the tainted history that you just went to the trouble of purging.

2. After some time has passed and you're confident that git filter-branch had no unintended side effects, you can force all objects in your local repository to be dereferenced and garbage collected with the following commands (using Git 1.8.5 or newer):

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

   git reflog expire --expire=now --all

   git gc --prune=now

## Examples

### Using filter-branch

```
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
```

Add your file with sensitive data to .gitignore to ensure that you don't accidentally commit it again.

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

Push your local repo to GitHub

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

In order to remove the sensitive file from your tagged releases, you'll also need to force-push

against your Git tags:

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

### Using the BFG Repo Cleaner

BFG Repo cleaner is an alternative to git filter-branch. It can be used to remove sensitive data or large files that were committed wrongly like binaries compiled from the source. It is written in Scala.

Project website: BFG Repo Cleaner

## Requirements

The Java Runtime Environment (Java 7 or above - BFG v1.12.3 was the last version to support Java 6). The Scala library and all other dependencies are folded into the downloadable jar.

## Remove files with sensitive data

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

Read Removing sensitive data or large files online:
https://riptutorial.com/github/topic/8170/removing-sensitive-data-or-large-files

# Chapter 12: Update a forked Repository

## Remarks

- GitHub Help: Configuring a remote for a fork
- GitHub Help: Syncing a fork
- popular ans in StackOverFlow

## Examples

### Config a remote for your fork then sync your fork (master branch)

1. Config a remote for my fork

```
$ 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. Sync my fork locally

```
$ 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. Sync my fork on Github

```
$ git push origin master
```

Read Update a forked Repository online: https://riptutorial.com/github/topic/3758/update-a-forked-repository

# Chapter 13: Using Gist

## Introduction

Gists are a great way to share your work. You can share single files, parts of files, or full applications. You can access gists at https://gist.github.com.

Every gist is a Git repository, which means that it can be forked and cloned. The gist editor is powered by CodeMirror.

There are two types of gists: public gists and secret gists.

Additionally, if you are not logged into GitHub when you create your gist, it will be an anonymous gist.

## Remarks

Gists are a great way to share your work. You can share single files, parts of files, or full applications.

There are two types of gists: public gists and secret gists. Additionally, if you are not logged into GitHub when you create your gist, it will be an anonymous gist.

**Public Gists**

Public gists show up in Discover, where people can browse new gists as they're created. They're also searchable, so you can use them if you'd like other people to find and see your work.

**Secret Gists**

Secret gists don't show up in Discover and are not searchable. Use them to jot down an idea that came to you in a dream, create a to-do list, or prepare some code or prose that's not ready to be shared with the world.

You can create as many secret gists as you like.

**Anonymous Gists**

If you create a gist without logging into GitHub, it will be an anonymous gist. Anonymous gists can be public or secret. To delete an anonymous gist on GitHub.com or GitHub Enterprise, contact GitHub support or your site administrator, respectively. Please provide the URL of the gist you wish to delete.

## Examples

### Public Gist

A public gist can be *almost* anything.

A simple example of a Javascript function:

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

## Secret Gist

A secret gist should be used for anything that you don't want to appear publicly on GitHub. Secret gists can be used when you don't want private keys to be accessible to the public, or for and private code in general.

A simple example of JSON code that would be better fit for a secret gist:

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

Read Using Gist online: https://riptutorial.com/github/topic/7978/using-gist

# Chapter 14: Using GitHub Buttons

## Introduction

What are GitHub buttons? GitHub buttons are buttons that you can add to your website that redirects users to any repository that you like!

## Remarks

Credits:

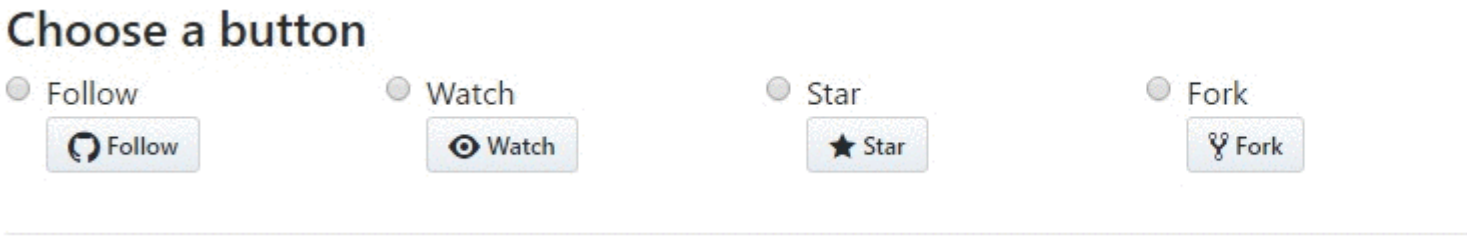
- Gif images recorded with Recordit
- Static images taken with Snipping Tool
- Code editor used in full tutorials was codepen.io

## Examples

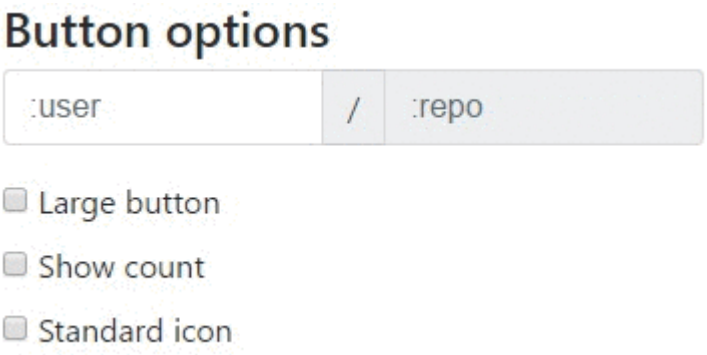
### Follow Button

A follow button is a button that links to a GitHub user page and prompts the user to follow the user. Here's how to create one:

1. Go onto github:buttons
2. Click "Follow"




3. Place your GitHub username in the box labeled ":user"



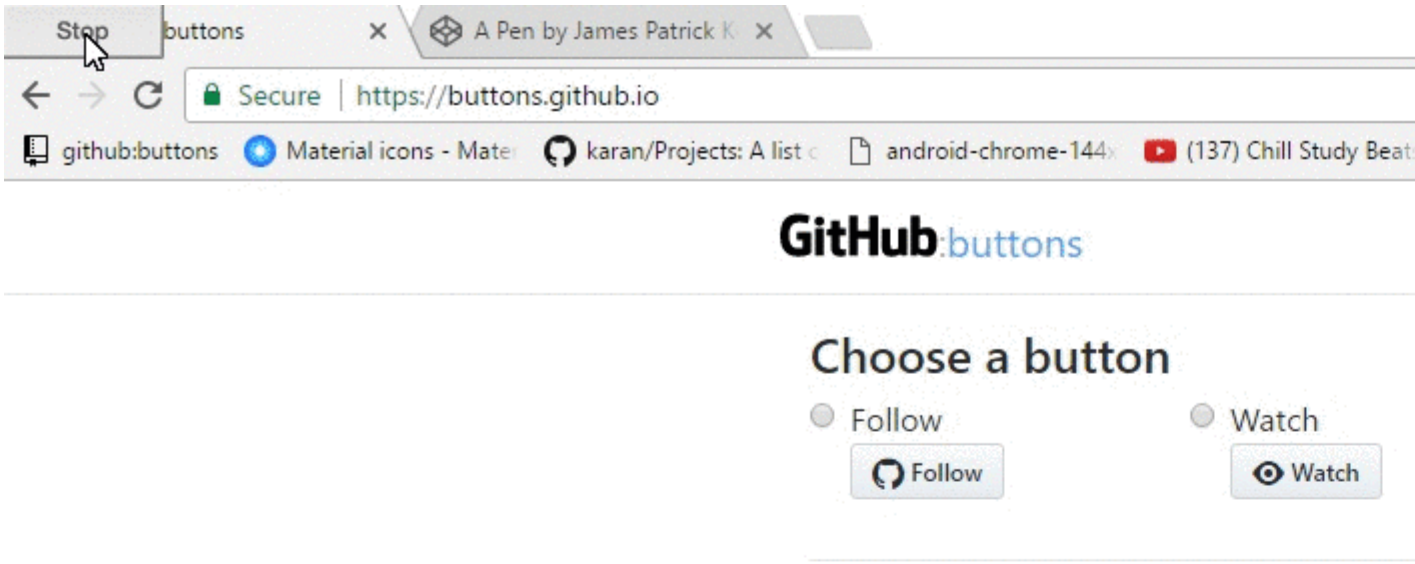

4. Customize the button using the boxes "Large button", "Show count", and "Standard icon":




5. Place this code in the `<head>` or before the end of the `<body>` of your code:

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

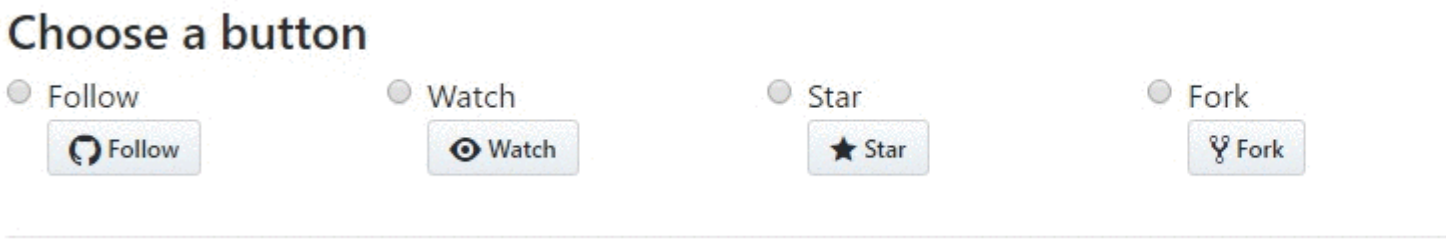
6. Place the customized button rendering code in your code.


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

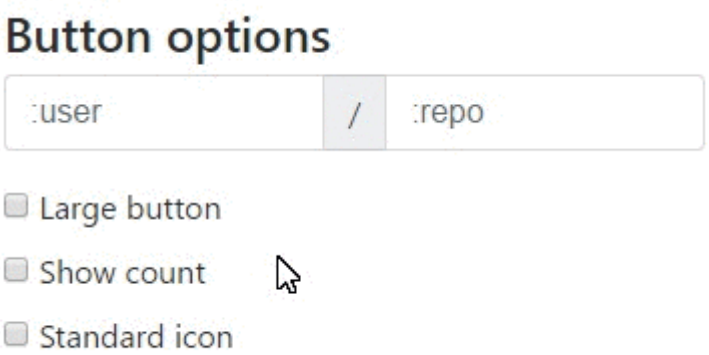
- **Star** a repository
- **Fork** a repository
- **Download** a repository
- List an **Issue** with a repository

Here's how to create some:

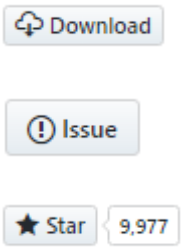
1. Go onto github:buttons
2. Click the button type you want to create (Watch, Star, Fork, Download, or Issue)




3. Place your GitHub username in the box labeled ":user", and your repository in the box ":repo"



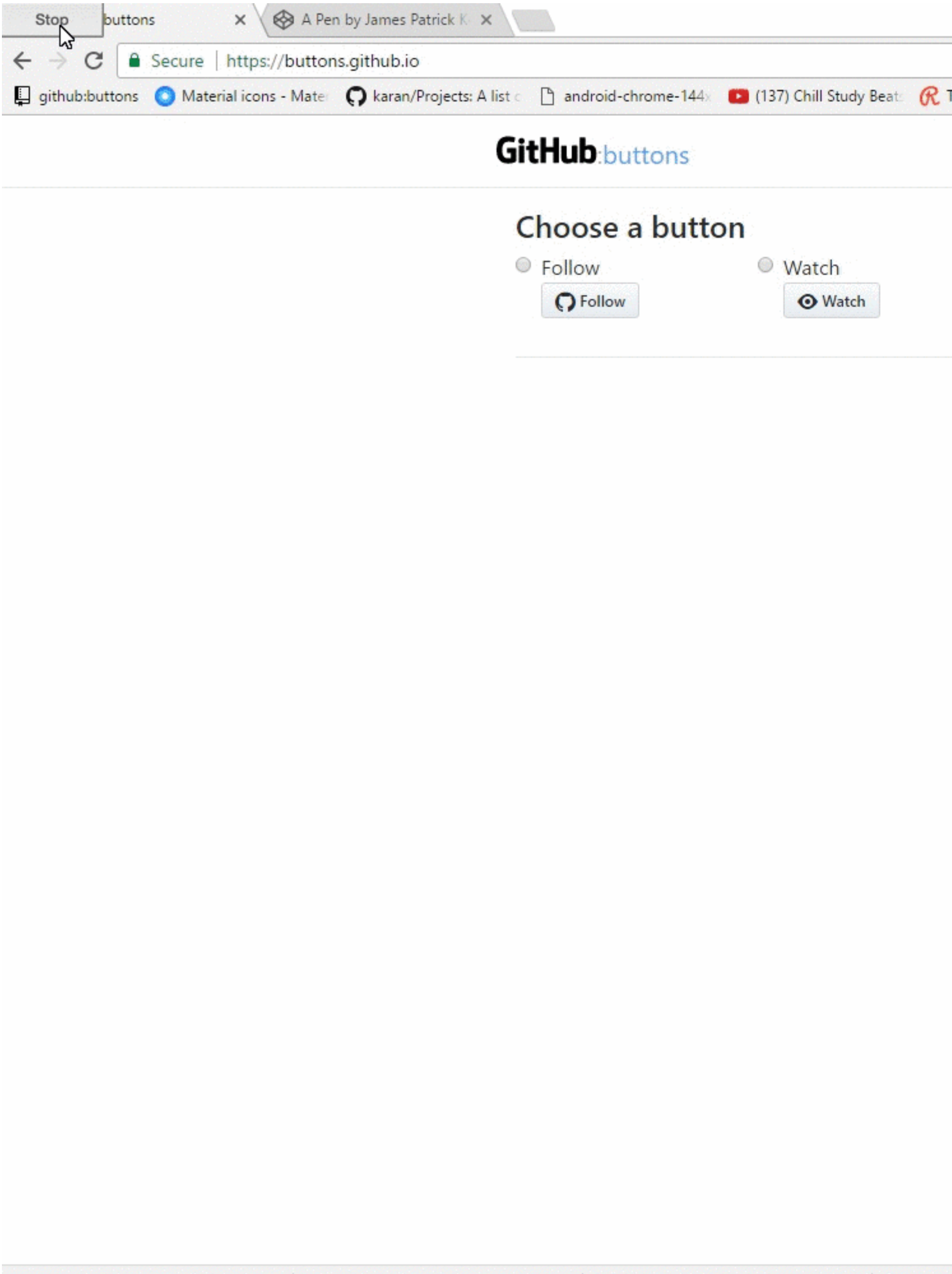

4. Customize the button using the boxes "Large button", "Show count", and "Standard icon":




5. Place this code in the `<head>` or before the end of the `<body>` of your code:

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

6. Place the customized button rendering code in your code.


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

# Chapter 15: Working with Gitflow

## Syntax

- git flow <subcommand>
- git flow init
- git flow [feature|release|hotfix] [start|finish]

## Parameters

| Subcommand | Details |
|---|---|
| init | Initialize a new git repo with support for the branching model. |
| feature | Manage your feature branches. |
| release | Manage your release branches. |
| hotfix | Manage your hotfix branches. |

## Remarks

- gitflow concept from author
- branch model picture

## Examples

### Operation on 5 common branches locally

One of most common use cases of Gitflow

1. *Initialize* repo and define branches

```
$ 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. *Start* and *Finish* a Feature

```
$ 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. *Start* and *Finish* a Release

```
$ 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. *Start* and *Finish* a Hotfix

```
$ 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'
```

Read Working with Gitflow online: https://riptutorial.com/github/topic/6231/working-with-gitflow

# Credits

| S. No | Chapters | Contributors |
|---|---|---|
| 1 | Getting started with github | BadAllOff, BrechtDeMan, Community, H. Pauwelyn, Hamzawey, Hugo, intboolstring, Kronos, Mateusz Piotrowski, Minhas Kamal, Nicholas Qiao, rpadovani |
| 2 | Backing up GitHub | geek1011 |
| 3 | Cloning a repository from GitHub | demonplus, geek1011, Hamzawey, James Kerrane, Mateusz Piotrowski |
| 4 | Displaying GitHub timeline / feeds in your Website | Hugo, Newton Joshua |
| 5 | download single file from GitHub repository | ownsourcing dev training |
| 6 | GitHub Desktop | creyD |
| 7 | GitHub Pages | BrechtDeMan, geek1011, Mono |
| 8 | How to create custom GitHub Labels? | Ahmad Awais |
| 9 | Issues | geek1011, Hamzawey, SuperBiasedMan |
| 10 | Pull Requests | Maxcell |
| 11 | Removing sensitive data or large files | Gautam Krishna R, Kronos |
| 12 | Update a forked Repository | Derek Liu |
| 13 | Using Gist | Kronos, tehp |
| 14 | Using GitHub Buttons | James Kerrane |
| 15 | Working with Gitflow | Derek Liu |