GitHub

Succinctly®

by Joseph D. Booth

Syncfusion® | Technology Resource Portal

GitHub Succinctly

By

Joseph D. Booth

Foreword by Daniel Jebaraj

Technical Reviewer: James McCaffrey
Copy Editor: Courtney Wright
Acquisitions Coordinator: Morgan Weston, social media marketing manager, Syncfusion, Inc.
Proofreader: Darren West, content producer, Syncfusion, Inc.

Table of Contents

The Story behind the *Succinctly* Series of Books

Daniel Jebaraj, Vice President
Syncfusion, Inc.

Staying on the cutting edge

As many of you may know, Syncfusion is a provider of software components for the Microsoft platform. This puts us in the exciting but challenging position of always being on the cutting edge.

Whenever platforms or tools are shipping out of Microsoft, which seems to be about every other week these days, we have to educate ourselves, quickly.

Information is plentiful but harder to digest

In reality, this translates into a lot of book orders, blog searches, and Twitter scans.

While more information is becoming available on the Internet and more and more books are being published, even on topics that are relatively new, one aspect that continues to inhibit us is the inability to find concise technology overview books.

We are usually faced with two options: read several 500+ page books or scour the web for relevant blog posts and other articles. Just as everyone else who has a job to do and customers to serve, we find this quite frustrating.

The *Succinctly* series

This frustration translated into a deep desire to produce a series of concise technical books that would be targeted at developers working on the Microsoft platform.

We firmly believe, given the background knowledge such developers have, that most topics can be translated into books that are between 50 and 100 pages.

This is exactly what we resolved to accomplish with the *Succinctly* series. Isn't everything wonderful born out of a deep desire to change things for the better?

The best authors, the best content

Each author was carefully chosen from a pool of talented experts who shared our vision. The book you now hold in your hands, and the others available in this series, are a result of the authors' tireless work. You will find original content that is guaranteed to get you up and running in about the time it takes to drink a few cups of coffee.

Free forever

Syncfusion will be working to produce books on several topics. The books will always be free. Any updates we publish will also be free.

Free? What is the catch?

There is no catch here. Syncfusion has a vested interest in this effort.

As a component vendor, our unique claim has always been that we offer deeper and broader frameworks than anyone else on the market. Developer education greatly helps us market and sell against competing vendors who promise to "enable AJAX support with one click," or "turn the moon to cheese!"

Let us know what you think

If you have any topics of interest, thoughts, or feedback, please feel free to send them to us at succinctly-series@syncfusion.com.

We sincerely hope you enjoy reading this book and that it helps you better understand the topic of study. Thank you for reading.

Please follow us on Twitter and "Like" us on Facebook to help us spread the word about the *Succinctly* series!

GitHub Succinctly

What exactly is GitHub? Many Linux users are familiar with Git (a version control tool), and most Windows developers use version control, such as Subversion. Version control is a useful tool to manage a set of files to track changes over a period of time. It allows users to compare file changes, restore prior versions, merge conflicting versions together, and more. It does this by storing the file versions in a repository database. You can check files in and out of the repository, compare versions (say between your version and the current repository), etc.

GitHub, in a gross simplification, is a version control repository that is stored on the web. There is much more to it than that, however. Imagine social media and repositories together, and you begin to see what GitHub is all about. As of 2015, there are over 11 million GitHub users and almost 30 million repositories.

In this book, we are going to cover creating an account and searching those 30 million repositories. But exploring the works of millions of GitHub projects (including projects by Google, Yahoo, and Microsoft, to name a few) is just part of the fun. GitHub also lets you create and update your own repositories, which you can share with other users. Developers from all over the world can also work on copies of your repositories. Imagine having access to a large contingent of developers who might want to contribute. You can contribute to projects that spark your interest as well.

Of course, you can also create private repositories if you want to use the GitHub platform for source code development that you don't want to share publically. Many companies use the benefits of GitHub for their own private development repositories. Private repositories require a fee, but the fee is quite reasonable considering all of the benefits GitHub offers.

Note: The GitHub system is a large application. This book is aimed towards developers who want to search repositories, create or manage their own work, and collaborate with others. More advanced topics, like organization-level repositories and storing GitHub on internal servers, are not addressed in this book.

Chapter 1 Git – A Brief Overview

Although this book is focused on GitHub, there are some basic concepts and terms from Git that will be helpful to understand as you learn GitHub.

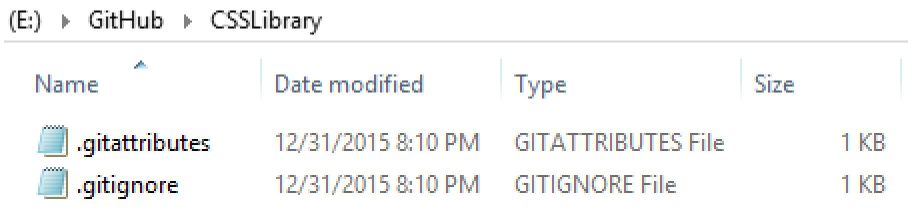
Working directory

The working directory is a folder on your computer or on a network share. Each developer has a complete copy of all of the code and associated files. You can do all your development work in the working directory, and let Git manage the version control aspects for that folder.

Git refers to this working directory as a repository, or a repos for short. When Git manages the folder, you get all the version control benefits that Git offers.

Git files

Within the working directory, you might find some extra files that are used to provide instructions to Git about the files in the folder. These are **`.gitattributes`** and **`.gitignore`**. The attribute file tells Git how to handle various files for comparison operations. The ignore file instructs Git that certain files (based on extension) should not be handled by Git. These files will be covered in more detail in a later chapter.


(E:) ▸ GitHub ▸ CSSLibrary

Name	Date modified	Type	Size
.gitattributes	12/31/2015 8:10 PM	GITATTRIBUTES File	1 KB
.gitignore	12/31/2015 8:10 PM	GITIGNORE File	1 KB

Figure 1: Starting Git empty working directory

When you create a repository, Git will add default versions of these files to your working directory. You can create a Git repository on an existing directory or have Git create a new repository (and working folder) for you. You can also clone an existing repository. A clone is a complete copy (including history) of the repository. No matter how you create it, you will now have a Git repository to start working in.

Clone

A clone is a complete copy of a repository. For example, you might make a copy of the repository, fix a few bugs, and then commit the code back to the repository. This assumes that you have write access to the repository however.

Fork

A fork is a copy of the repository created under your account for repositories that you cannot directly update. You can make your code changes in your copy, and then submit a pull request, asking your changes to be integrated back into the main repository. This is often the case on open source projects, where many collaborators help the author with issues, but the author decides when to include them in the main branch.

Pull request

A pull request is a request to the repository owner to merge your proposed changes back into the code base. This is usually done by a collaborator who has fixed a reported issue.

> ***Note: Pull/push terminology can be confusing. "Push" typically means moving information from a central location to subscribers, while "pull" means the subscribers request the information. In GitHub, the "subscriber" is the repository owner, who will want to pull changes from your (developer) local repository. If you view the owner as a subscriber, the definition of "pull request" means: "Hey, subscriber, I have some changes you might like…"***

Branches

A branch is a complete set of the repository files. Git creates a master branch by default. You can create additional branches to test out development ideas without impacting the master branch. Git allows you to compare code between branches, and at some point, to merge code from the "dev" branch back into the master branch. This allows multiple collaborators to work on a project without interfering with the base code.

Common branches might include:

- Feature branch: A branch dedicated to adding new features to the code base
- Release branch: The branch with code planned for next release
- Master branch: The default branch that all code eventually gets merged to
- Dev branch: The active development branch, where code is being work on

It's up to you as to how to create branches, or even if you want to. However, if many collaborators are working on project, branches make it easier to develop enhancements without losing the ability to build a release from the main branch.

Tracked files

Tracked files are files in the working directory (repository) that Git manages. As you add new files or make updates to existing files, Git tracks these changes. At some point, you will want to add these files to the repository via the **`commit`** command.

Commit

Commit is the process of adding any changes in tracked files to repository. You will enter a summary and description, then add the files to the repository. Git will maintain a history of all commits against a repository.

Commit message

The commit message is a summary and description applied to the current set of files being committed to the repository.

Merging

Merging is the process of combining code from one branch to another. After the fixes and enhancements in a branch are completed, you will typically want to merge the code back into the master branch. Git provides differential tools to show the code changes and assist with the merge process.

Change sets

Git allows you to view all the changes you've made compared to the branch you are using. You can review the changes and decided which file changes you would like to commit to the branch. For example, you might have completed work on two out of five files you've changed, and you'd like to commit them separately. Git will report changes in all five files. You commit the two files, and Git adds them to history, while still tracking the other files. This allows you to create logical change sets, rather than simply date-based changes.

Commit history

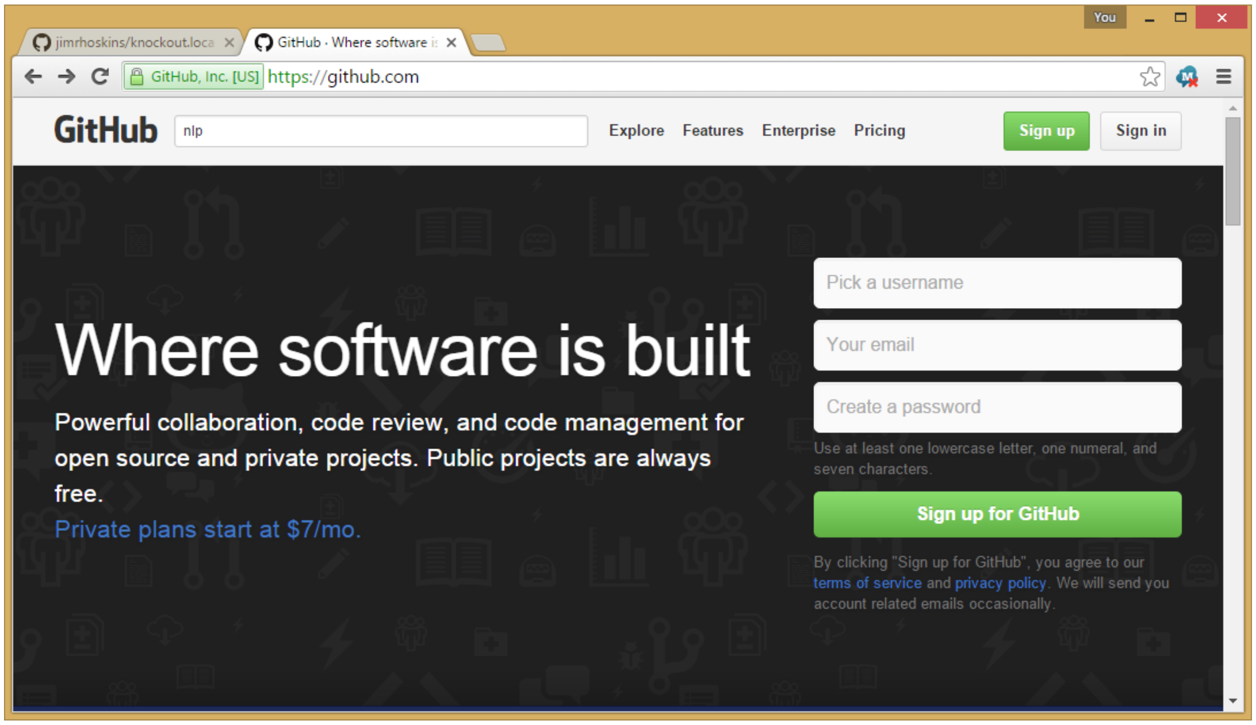
Git keeps a history log of all committed changes, so you can review the changes made in any prior commits. If you've cloned a repository, you will have the commit history from that repository as well.

Summary

The basic Git flow is to create a repository, and make your changes and additions. Then, commit as many change-sets as you'd like. Git will report changes to your tracked file and allow you to decide how to organize your commits. Once you've done a number of commits, you can use Git's commit history to view project status.

Chapter 2 Getting Started

The best way to get started with GitHub is to open your browser and jump right in. Open your favorite browser and enter https://github.com/ to open the GitHub main page. You will see a screen similar to the following:



Where software is built

Powerful collaboration, code review, and code management for open source and private projects. Public projects are always free.

Private plans start at $7/mo.

Use at least one lowercase letter, one numeral, and seven characters.

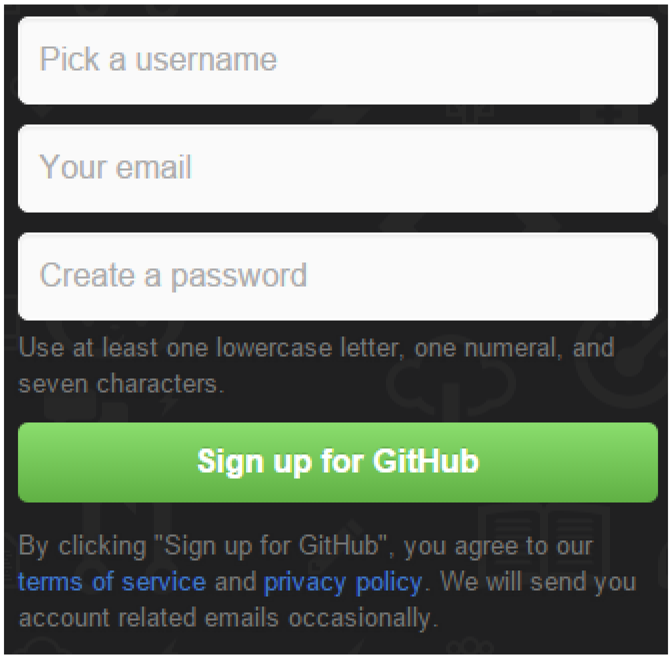
By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.

Figure 2: GitHub home page

While you can search GitHub repositories without creating an account, this chapter covers the basics of setting up an account. If you'd prefer to explore without setting up an account, you can jump ahead to Chapter 4, "Searching GitHub." However, the chapters on creating and managing your own repositories will require a GitHub user account.

Note: GitHub may change the look of the home page, so your initial page might not look exactly like the screenshot from Figure 2. Signing up

You can sign up for GitHub directly from the home page simply by filling in the requested user name, email address, and password. The fields are dynamically validated, reporting any issues as you type.



Use at least one lowercase letter, one numeral, and seven characters.

By clicking "Sign up for GitHub", you agree to our terms of service and privacy policy. We will send you account related emails occasionally.

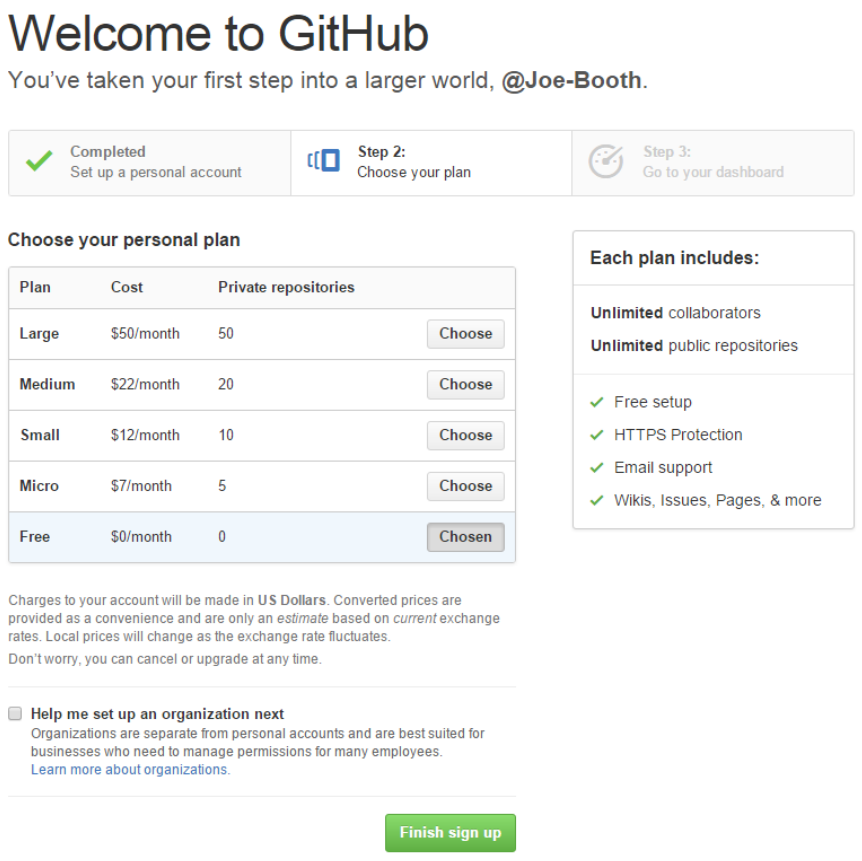
Figure 3: GitHub signup

Tip: Your user name will be visible on your public repositories, so be sure to choose one that you don't mind sharing with other people.

You will be able to change your user name from your profile screen, and GitHub will automatically update your repository information. However, if people have links to your old profile page, these will not be redirected to your new name, but rather return a 404 (not found) error. If you do change your user name, be sure to update any public profiles to reference the new name.

Choosing your plan

Once you've filled in your account information, the next step is to choose a GitHub plan. The default plan is the Free plan. This plan allows you to create public repositories and to interact with other repositories as well.


Welcome to GitHub

You've taken your first step into a larger world, **@Joe-Booth**.

Completed — Set up a personal account | Step 2: Choose your plan | Step 3: Go to your dashboard

Choose your personal plan

Plan	Cost	Private repositories	
Large	$50/month	50	Choose
Medium	$22/month	20	Choose
Small	$12/month	10	Choose
Micro	$7/month	5	Choose
Free	$0/month	0	Chosen

Charges to your account will be made in **US Dollars**. Converted prices are provided as a convenience and are only an *estimate* based on *current* exchange rates. Local prices will change as the exchange rate fluctuates.

Don't worry, you can cancel or upgrade at any time.

Each plan includes:

Unlimited collaborators

Unlimited public repositories

- ✓ Free setup
- ✓ HTTPS Protection
- ✓ Email support
- ✓ Wikis, Issues, Pages, & more

☐ **Help me set up an organization next**
Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.
Learn more about organizations.

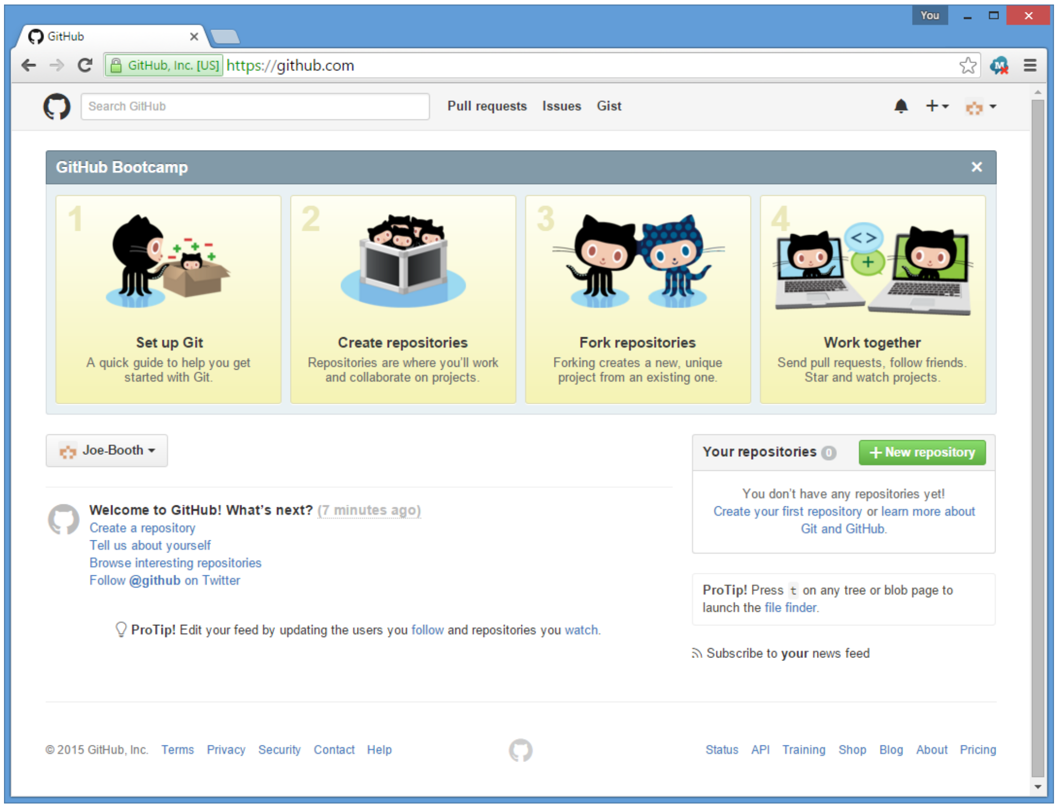
Figure 4: GitHub plan choices

The personal plans indicate the number of private repositories you plan on having. If you only plan on searching and using public repositories, the Free plan will meet your needs. However; as you work with GitHub, you may want to use it for your internal or private development work. In this case, the paid plans are reasonably priced.

Note: in this book we are focused on personal GitHub plans. There are also plans for organizations, which allow different users and different access based on teams. In addition, billing can be sent to a separate email account and team owners can access organization members.

Your dashboard

After entering basic account information and choosing a plan, you are taken to your personal dashboard page. The GitHub Bootcamp banner across the top lets you use a wizard approach to set up Git and your own repositories.



GitHub Bootcamp

1 Set up Git
A quick guide to help you get started with Git.

2 Create repositories
Repositories are where you'll work and collaborate on projects.

3 Fork repositories
Forking creates a new, unique project from an existing one.

4 Work together
Send pull requests, follow friends. Star and watch projects.

Welcome to GitHub! What's next? (7 minutes ago)
Create a repository
Tell us about yourself
Browse interesting repositories
Follow @github on Twitter

ProTip! Edit your feed by updating the users you follow and repositories you watch.

You don't have any repositories yet!
Create your first repository or learn more about Git and GitHub.

ProTip! Press t on any tree or blob page to launch the file finder.

Subscribe to your news feed

© 2015 GitHub, Inc.

Figure 5: GitHub Dashboard

At this point, you have a GitHub account set up, and you are able to search GitHub, download code (and other files), and interact with other user's repositories. However, you need to do additional setup work to create your own repositories. This includes installing Git on your computer and creating a Git repository. This is covered in the next chapter. If you are only interested in searching existing work, you can skip that chapter. You will need to come back to it if you decide to put your own work into GitHub.

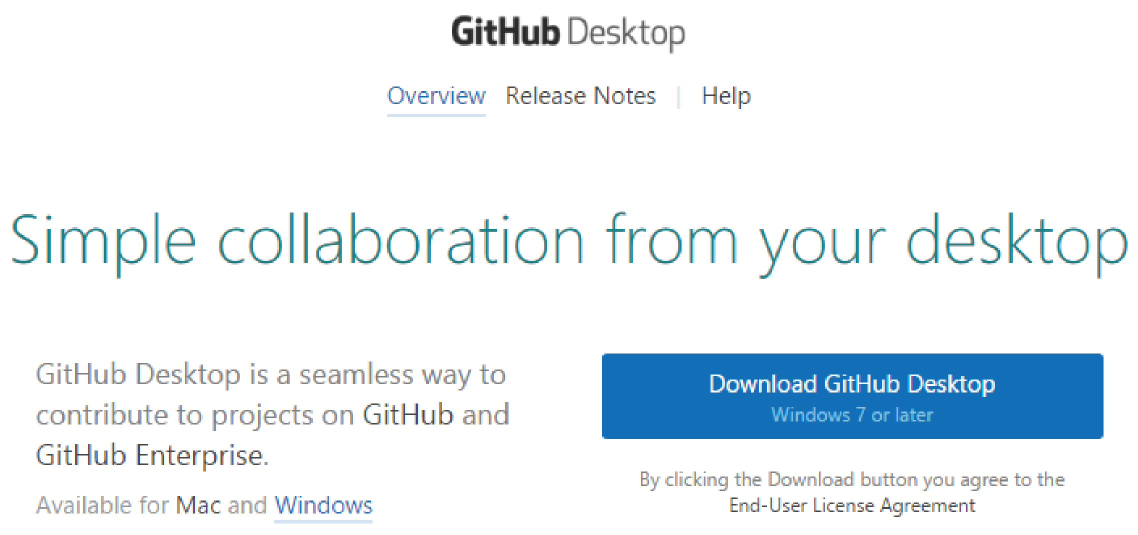
Chapter 3 Installing Git

In order to add your own projects to GitHub, you will need to create a local repository (just a folder) on your computer. You will then use the Git software to copy this folder's contents to a GitHub server. Although this book is focused on GitHub, some brief understanding of Git is necessary to work with GitHub repositories.

Tip: If you are interested in learning more about Git, be sure to download Ryan Hodson's e-book, Git Succinctly***.***

Downloading GitHub Desktop

The first step to installing Git on your local computer is to visit the GitHub Desktop website and download the desktop application.



Simple collaboration from your desktop

GitHub Desktop is a seamless way to contribute to projects on GitHub and GitHub Enterprise.

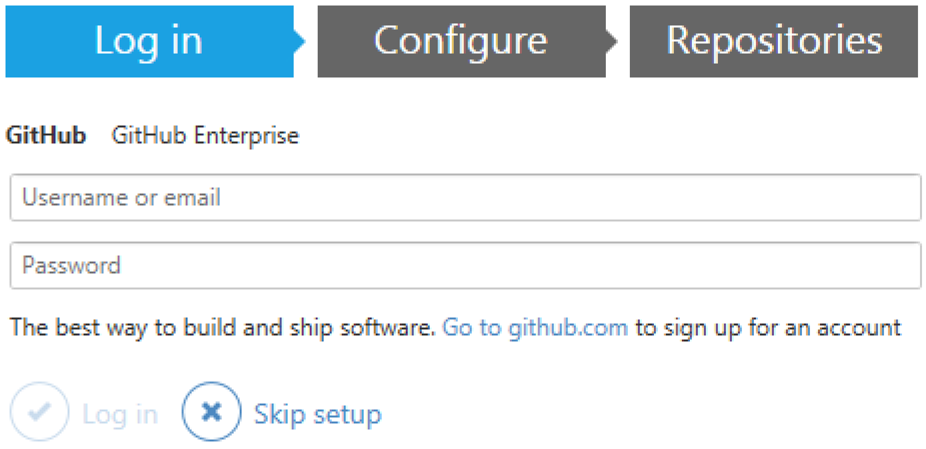
Available for Mac and Windows

By clicking the Download button you agree to the End-User License Agreement

Figure 6: GitHub Desktop site

Configuring GitHub Desktop

Once you've completed the installation, you'll need to link your desktop version to your GitHub account using the **Log in** tab.



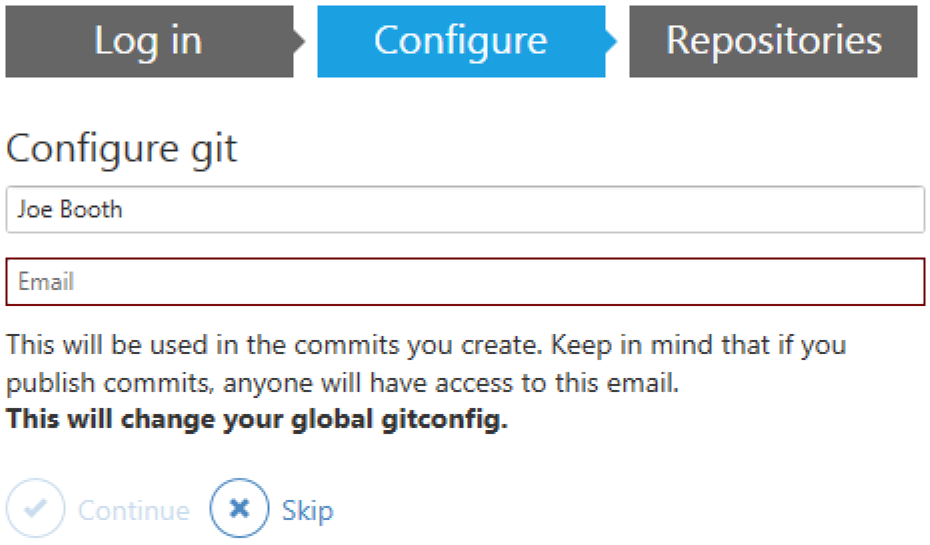
The best way to build and ship software. Go to github.com to sign up for an account

Figure 7: Desktop configuration

Once you've logged in, you move to the Configure tab.

Configure tab

The Configure tab is used to associate your account with any commits (writes to your repository) that you make.



Configure git

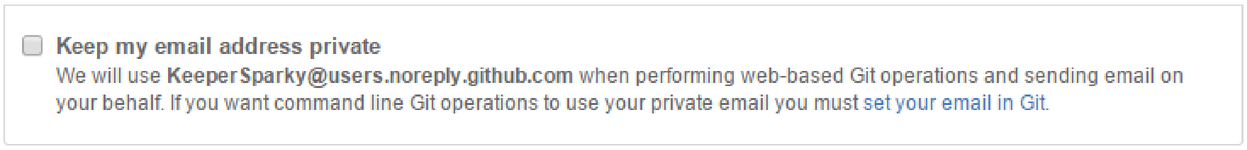
Joe Booth

Email

This will be used in the commits you create. Keep in mind that if you publish commits, anyone will have access to this email.
This will change your global gitconfig.

Figure 8: Git configuration

Git links your user ID and email address to your commits, so anyone viewing your repository will see your email address. If you'd prefer to keep your email address private, you can do so by going to GitHub and editing your profile. Under **Personal Settings**, navigate to the Emails menu item.



Keep my email address private

We will use KeeperSparky@users.noreply.github.com when performing web-based Git operations and sending email on your behalf. If you want command line Git operations to use your private email you must set your email in Git.

Figure 9: Email profile configuration

If you check this box, any Git operations will be sent to a special GitHub account (*username*@users.noreply.github.com). You will also need to set this email address in the Configure Git screen.

Repositories

If this is your first time using Git, you'll have no repositories set up, so you'll need to create one. GitHub Desktop will detect that and jump to the repository page.

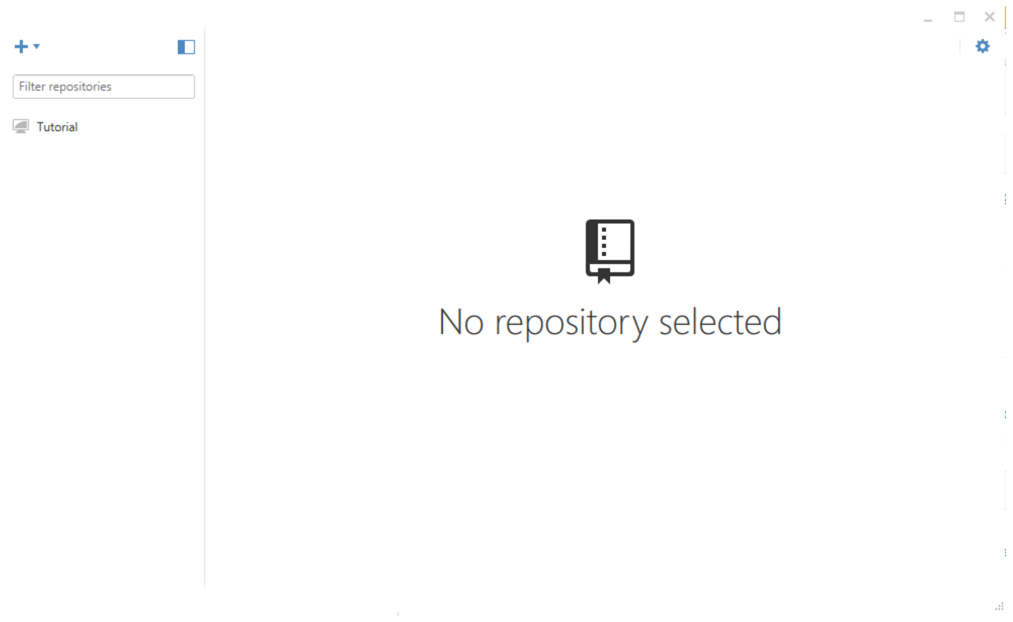
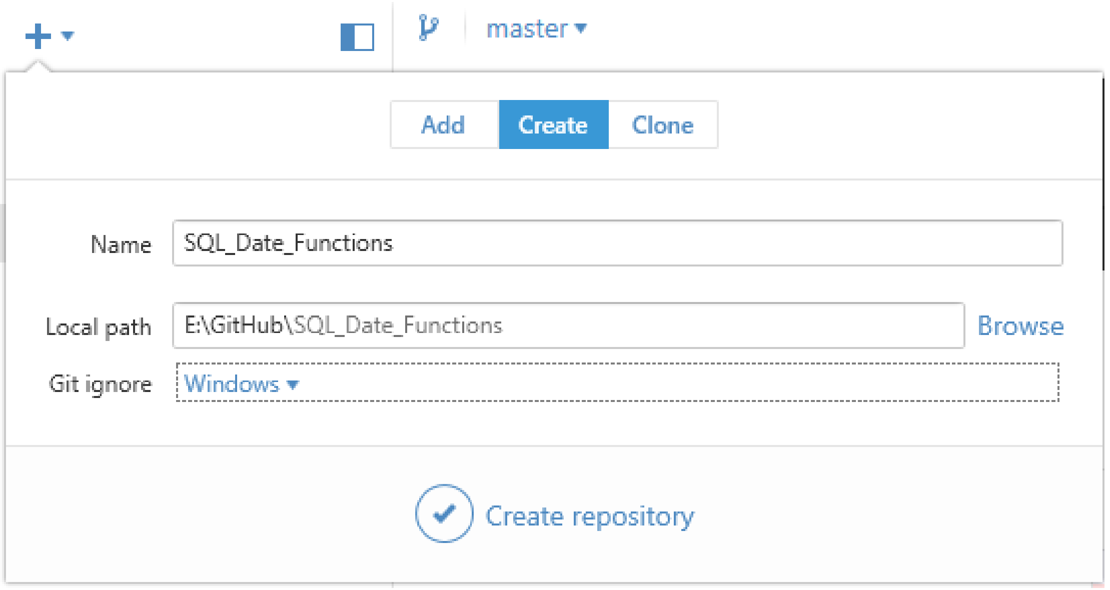


Figure 10: Repository page

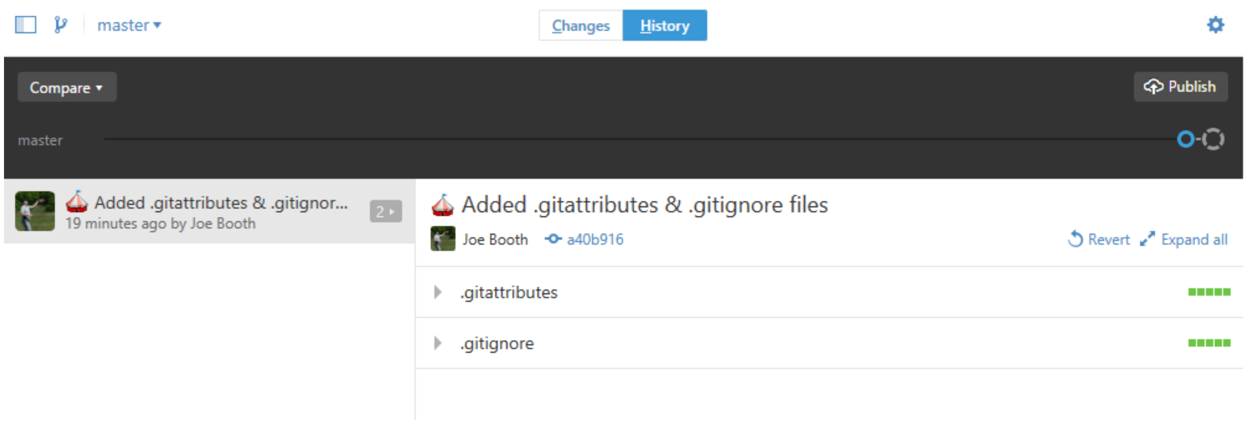
Click on the **+** (plus) sign to add a repository. By default, GitHub adds a folder called GitHub to your documents folder, which is where repositories will be stored. You can place your repositories anywhere by changing the local path when you first set up the repository. On the screen below, I am adding my SQL Date Functions repository to a local folder called E:\GitHub.



Name: SQL_Date_Functions
Local path: E:\GitHub\SQL_Date_Functions
Git ignore: Windows

Figure 11: Creating a repository

Once your repository is set up, you can move the files you want to this local folder. This will be the first step in publishing them to GitHub. When the repository is first set up, two files will be added, which basically provide configuration instructions to Git.



Added .gitattributes & .gitignor...
19 minutes ago by Joe Booth

Added .gitattributes & .gitignore files
Joe Booth a40b916

.gitattributes
.gitignore

Figure 12: Initial repository

Gitattributes

The default **.gitattributes** file provides a set of options controlling GitHub operations, particularly when comparing files. The list consists of a file extension (*.doc) and a command to perform when that file is detected. For example, the following line instructs GitHub to convert any file with a .PDF extension to plain text when running a comparison:

```
*.pdf  diff=astextplain
```

You can add other attributes to file extensions as well. For example:

```
*.jpg binary
```

This line tells Git that any file with a .JPG extension is a binary file. While Git is pretty good about understanding file types, you may have some files in your project that should be treated as binary (never compared or merged).

The .gitattributes file is optional; you can delete it if you don't need any special handling of the files in your repository (for example, only add source code files).

GitIgnore

The .**gitignore** file tells GitHub that certain files it might find should be ignored for tracking purposes. For a Windows system, you'll likely want to ignore such files as MSI installer, .DB image caches, etc. For example, the following line instructs Git to not track any MSI installer file in the repository:

```
*.msi
```

There are a number of matching rules, negating rules, etc. that you can add to the GitIgnore files for instructions to Git.

Creating a readme.md file

Once you've created the repository, you should create a **readme.md** text file. (The .md extension stands for Markdown Document). This file is displayed when your repository is opened on GitHub and can be searched by people using GitHub. Although it is a simple text file, there are some formatting options you can use. GitHub uses the Markdown syntax for letting you create plain-text files, which will be displayed as HTML. Follow this link if you want to learn more about Markdown, or use the GitHub help system to see what codes are supported in the text.

For our simple readme file about the SQL Date functions, we will use the **##<*text*>##** to create H2-level headers and the **###** for H3-level headers.

```
##SQL Date functions##

A couple date functions written for SQL-Server (although could be easily
ported to other platforms).

###Holiday_list()###
```

A good readme.md file should entice the user to explore your repository. You could also add a link to your own site using the following syntax:

```
[Joe Booth Consulting] (http://www.joebooth-consulting.com)
```

After you've created the file, save it in your GitHub local repository folder.

Loading files

If you click on **Changes**, you should see the **readme.md** file you just added. If there are no files, you'll be informed that there are no local changes, and you'll be given the chance to open the folder (repository) in Windows explorer. Copy the files you want to add into the folder. Git will now show you the changes made in the folder (the new files). For example, I've created a repository to hold my holiday list SQL code and a readme.md file:

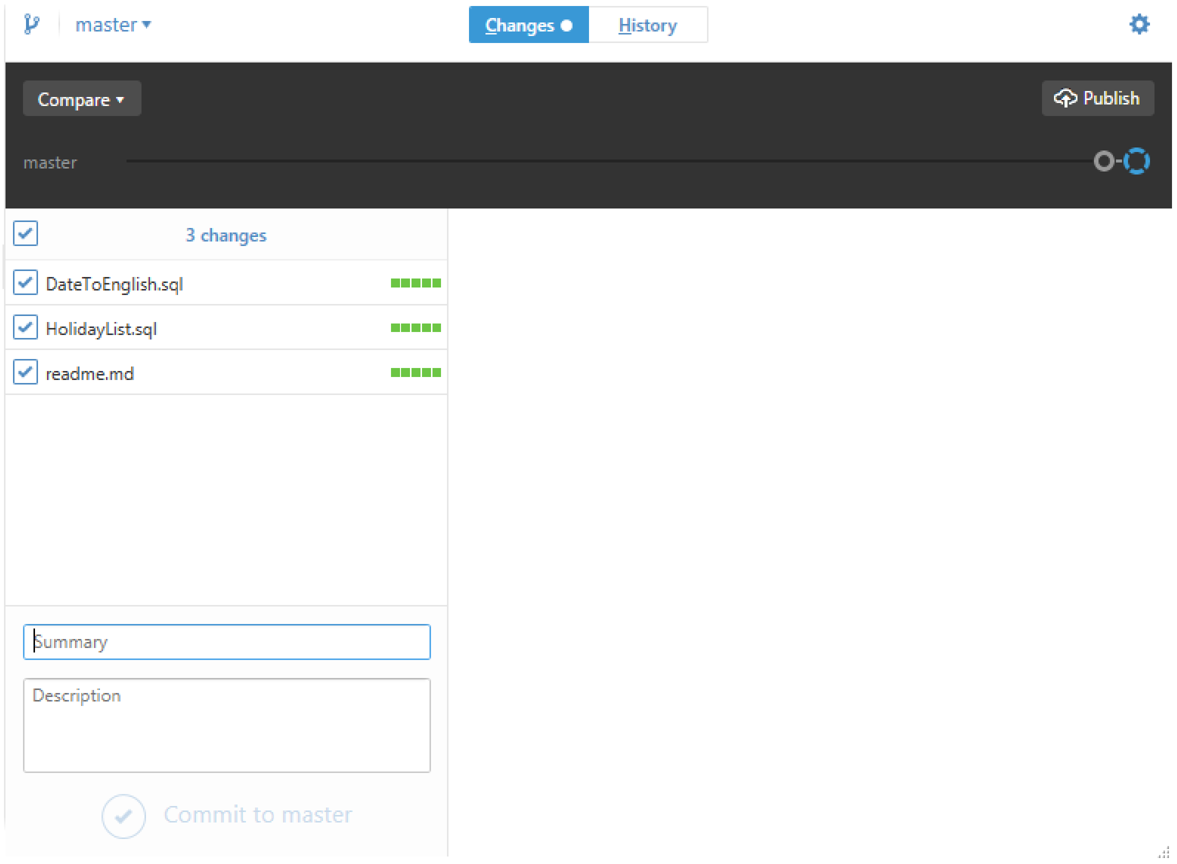
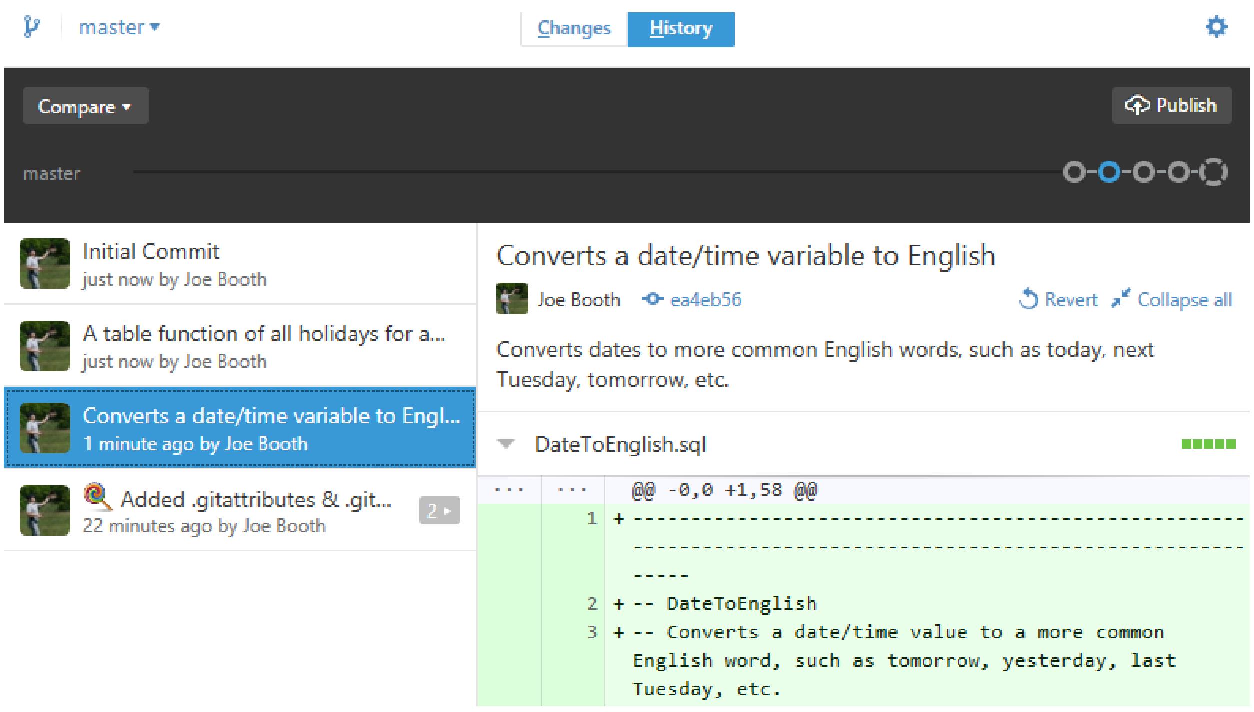


Figure 13: Initial set of changes

In order to commit these files to the local branch, you should enter a summary and a description. The summary is required to commit the changes to the master branch. However, it's a good idea to provide both a summary and description. Once you've done that, click the Commit to Master button.

Note: The summary and description apply to all files you've changed, so you may want to commit the files individually to give better descriptions for each file.

You can now go back to the **History** tab to see the set of changes made (the Git files and the files you've added).



Initial Commit
just now by Joe Booth

A table function of all holidays for a...
just now by Joe Booth

Converts a date/time variable to Engl...
1 minute ago by Joe Booth

Added .gitattributes & .git...
22 minutes ago by Joe Booth

Converts a date/time variable to English

Joe Booth ea4eb56 Revert Collapse all

Converts dates to more common English words, such as today, next Tuesday, tomorrow, etc.

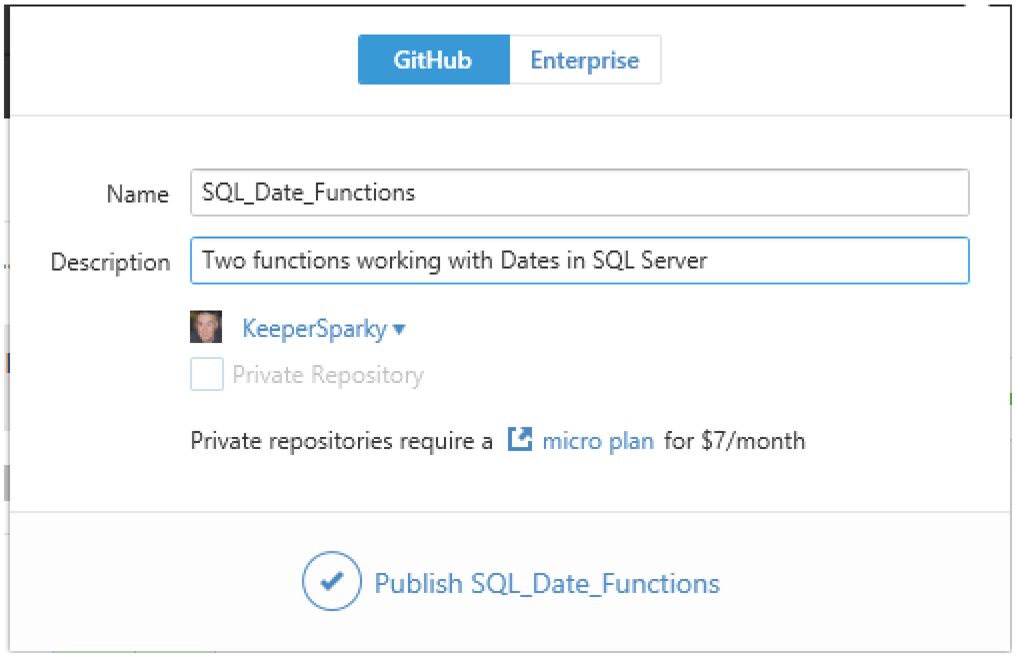
DateToEnglish.sql

```
@@ -0,0 +1,58 @@
+ -----------------------------------------------------------------------------------------------------------------
+ -- DateToEnglish
+ -- Converts a date/time value to a more common English word, such as tomorrow, yesterday, last Tuesday, etc.
```

Figure 14: Changes made to the repository

Publishing the repository

For the final step, click on the **Publish** button to publish the repository to the GitHub servers.



Name: SQL_Date_Functions

Description: Two functions working with Dates in SQL Server

KeeperSparky

Private Repository

Private repositories require a micro plan for $7/month

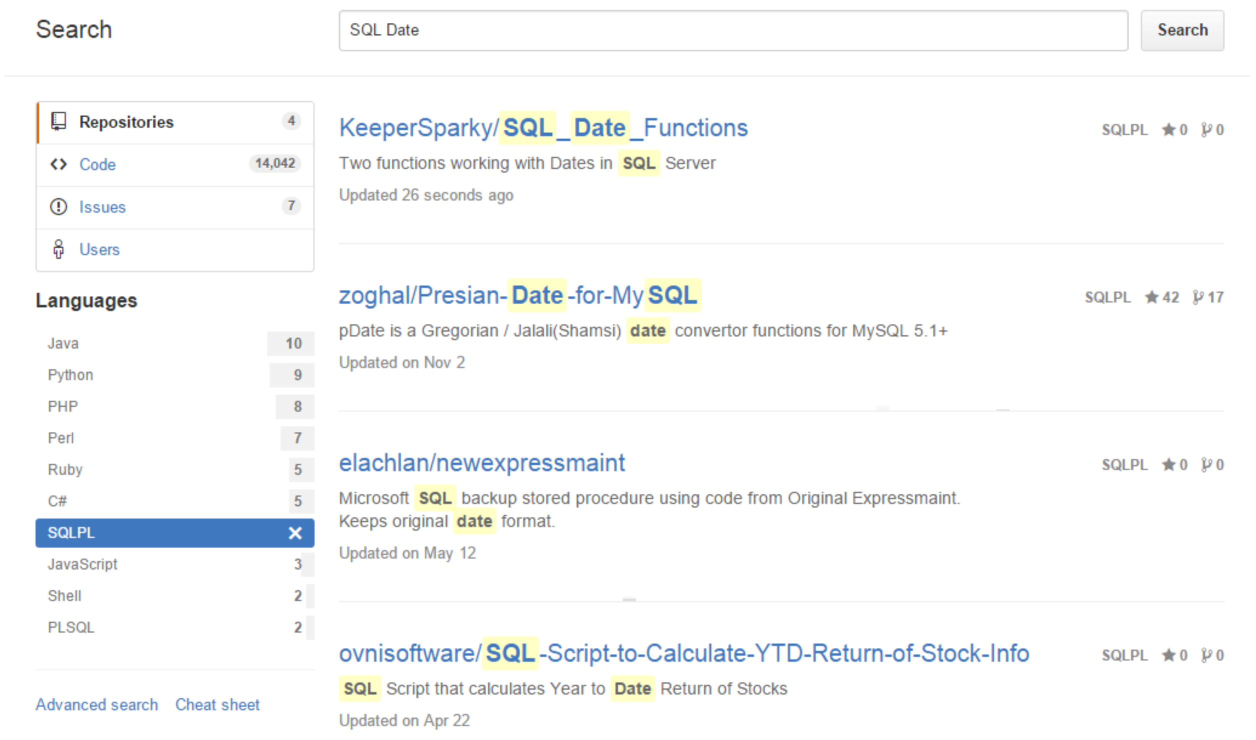
Publish SQL_Date_Functions

Figure 15: Publishing your repository

Congratulations—you've just published your first repository to GitHub.

Search for it

After publishing, a search of GitHub finds my repository.



Search

SQL Date

Search

Repositories 4
Code 14,042
Issues 7
Users

Languages

Java	10
Python	9
PHP	8
Perl	7
Ruby	5
C#	5
SQLPL	
JavaScript	3
Shell	2
PLSQL	2

Advanced search Cheat sheet

KeeperSparky/SQL_Date_Functions SQLPL ★0 0
Two functions working with Dates in SQL Server
Updated 26 seconds ago

zoghal/Presian-Date-for-MySQL SQLPL ★42 17
pDate is a Gregorian / Jalali(Shamsi) date convertor functions for MySQL 5.1+
Updated on Nov 2

elachlan/newexpressmaint SQLPL ★0 0
Microsoft SQL backup stored procedure using code from Original Expressmaint.
Keeps original date format.
Updated on May 12

ovnisoftware/SQL-Script-to-Calculate-YTD-Return-of-Stock-Info SQLPL ★0 0
SQL Script that calculates Year to Date Return of Stocks
Updated on Apr 22

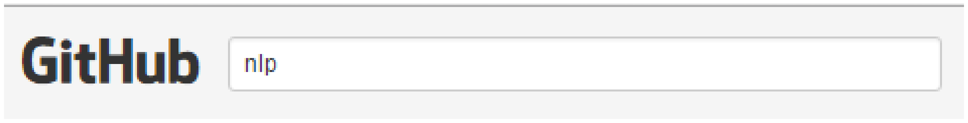
Figure 16: Finding your repository

Summary

This chapter is a brief introduction to installing GitHub Desktop and publishing your first repository to GitHub. Later in the book, we will cover interacting with the repository and collaborating with others on your code.

Chapter 4 Searching GitHub

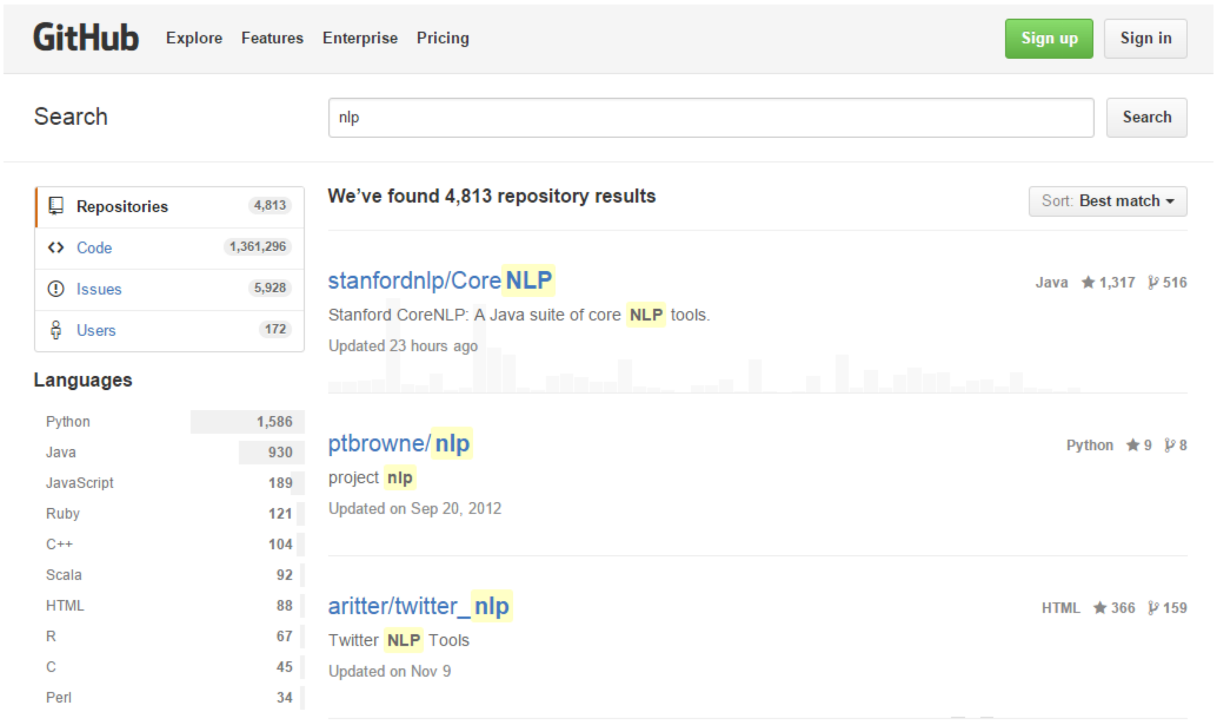
Searching GitHub is pretty simple; much like a Google search, you simply type what you're looking for in the search bar at the top of the screen.



nlp

Figure 17-Searching GitHub

Hit **Enter** to bring back the search results. In this example, I've searched for any repository about NLP (Natural Language Processing). For this particular search, GitHub found over 4,800 matching repositories—maybe a few more than I had hoped.



Search: nlp

Repositories	4,813
Code	1,361,296
Issues	5,928
Users	172

Languages

Python	1,586
Java	930
JavaScript	189
Ruby	121
C++	104
Scala	92
HTML	88
R	67
C	45
Perl	34

We've found 4,813 repository results

stanfordnlp/CoreNLP — Java ★ 1,317 516
Stanford CoreNLP: A Java suite of core NLP tools.
Updated 23 hours ago

ptbrowne/nlp — Python ★ 9 8
project nlp
Updated on Sep 20, 2012

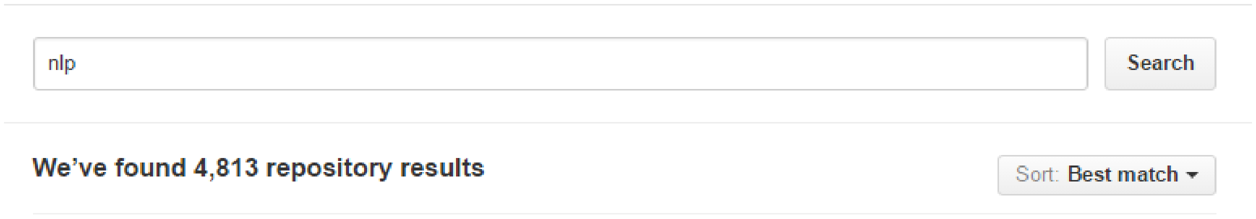
aritter/twitter_nlp — HTML ★ 366 159
Twitter NLP Tools
Updated on Nov 9

Figure 18: GitHub search results

Fortunately, GitHub provides both sorting and filtering options to make searching all those repositories easier.

Sorting the results

By default, GitHub returns the most likely matches to your search query. The Sort dropdown control below your search bar shows you the current sort order of the results:



We've found 4,813 repository results

Figure 19: Sort order

If you click on the Down Arrow key, you can see the other sort options. This will help you quickly explore the search results.

Most stars

As we mentioned earlier, GitHub offers social media-like features, in addition to simply being a repository. If users find a particular repository useful, or if they are actively involved in it, they can give the project a star. For example, the popular JavaScript Bootstrap framework has over 90,000 stars, and jQuery is approaching 40,000.

Note: You must have an account and be signed in to "star" a repository.

If you're a new developer, and want to know what the most popular JavaScript framework is, a search and sort by "Most Stars" would give you an idea of which framework might be a good starting point to learn.

blueimp/jQuery-File-Upload
JavaScript
22,281
5,914

Figure 20: Rating Stars

You can see the number of stars next to the repository name, as shown in Figure 20. As you can see, the File-Upload jQuery module from blueimp has over 22,000 stars. It's probably a safe bet that it's been tested and used by a large pool of developers.

Fewest stars

You can also sort by fewest stars, but you shouldn't assume that a project with fewer stars is necessary a bad project. The fact that it has less stars might also attest to the fact it's an unknown project or a new contribution. It could be a great project, but very few people either need it or have discovered it yet. There can be some hidden gems in these type of projects, particularly if you are working on an uncommon application.

Most/least forks

Although we will discuss GitHub forks in a later chapter, a brief description here should help. A fork is a copy of the repository that someone has downloaded, most likely with the intent to suggest changes or as a starting point for their own project.

Note: A fork isn't simply a download of the files; you can do that easily. A fork is generally made with the intent to add to the existing project.

The number of forks gives you a sense of how many developers are interested and contributing to the project. For example, there's a LINQ to CSV project that allows .NET developers to use LINQ Queries against CSV (Comma Separated Values) files. This project has 64 stars (as of this book) and 51 forks. This would indicate that the project has a lot of work being done on it, and suggests a number of developers are using the library.

Sometimes, a developer who needs some code might put a first version up on GitHub and find other interested developers to work on the project. Collaboration is a hallmark of the Open Source community, so an interesting project may get help from all over the world from a community of users to improve existing code.

A project with few forks, but lots of stars, could suggest a very stable and complete project. A project with many forks, but few stars, might be a niche topic that a small number of very active developers are using.

Recently updated

Another sort option is the recently updated sort, which sorts the list from most recently updated to oldest updates. New projects or recently active projects will be presented first. Simple downloads of the project don't count as activity, only actions that update the repository. Some projects don't necessarily need recent updates. For example, there is a .NET component to retrieve geographic coordinates (latitude and longitude) from a zip code. It hasn't been active in over a year, but the data content was based on the most recent (2010) census data, so once the code settles down, there's no need to update it until another census or more current zip code data becomes available.

Least recently updated

Least recently updated shows the oldest (in terms of activity) first. Again, the project update date only provides an idea of how active the project is. Scriptaculous (a JavaScript framework) hasn't been updated in six months, but has almost 2,400 stars. This suggests a framework that is still in use, and pretty mature. It was actually written in 2005 (ancient in Internet time) and feature-frozen in 2010, yet still is in use, and still has contributors listed on the project even though it's frozen.

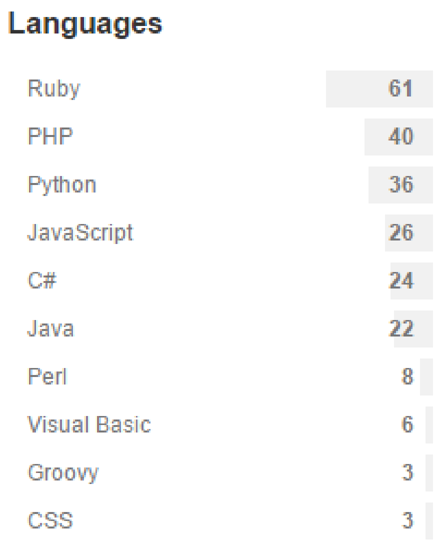
With new browsers and technology coming out frequently, an older web project might not be compatible with the new technology. Remember that I.E. 9 was not released until 2011, so this particular framework (Scriptaculous) worked well with I.E. 8 versions. It could be a good framework for web applications where the computers are older and not running the latest browsers.

Between the sort options, the stars (how many developers like the project after using it), the forks (how many developers are interested enough to enhance the project), and activity (how recently was the project worked on), you should be able to choose a good project for your application.

Filtering the results

In addition to sorting the results, the GitHub search results also provides some basic filtering options, primarily letting you focus on a particular programming language. For example, let's assume we are working on a project to integrate QuickBooks with some application you are working on. A search for "QuickBooks" in GitHub yields over 300 results. If I sort them by most stars, I find PHP and Ruby implementations dominating the top 10 results.

However, to the left side of the search results, GitHub shows a breakdown of languages groups for all the found repositories.



Languages

Ruby	61
PHP	40
Python	36
JavaScript	26
C#	24
Java	22
Perl	8
Visual Basic	6
Groovy	3
CSS	3

Figure 21: Languages

By clicking on the language name, we can filter the results to only display the 24 projects that are about QuickBooks and use C# as the base language.

Note: Keep in mind that a library written in C# can be accessed from any .NET language, such as Visual Basic.

Summary

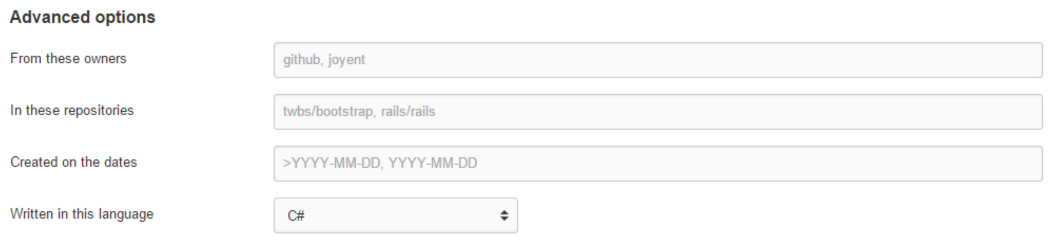
The basic searching within GitHub, and the sorting and filtering options, should allow you to move through the repositories and find the library, application, etc. that you need to jumpstart your development efforts.

Chapter 5 Advanced Searching

GitHub's search engine hides the advanced options from a search, but provides a lot of additional functionality using different constructs within your search string. You can explore the advanced options by clicking on the Advanced search link underneath the list of languages.

Advanced options

The Advanced options section allows you to look for repositories owned by particular users and companies, or even restrict the search to a particular repository. You can also search by date range or filter results to a particular programming language.



Advanced options	
From these owners	github, joyent
In these repositories	twbs/bootstrap, rails/rails
Created on the dates	>YYYY-MM-DD, YYYY-MM-DD
Written in this language	C#

Figure 22: Advanced options search

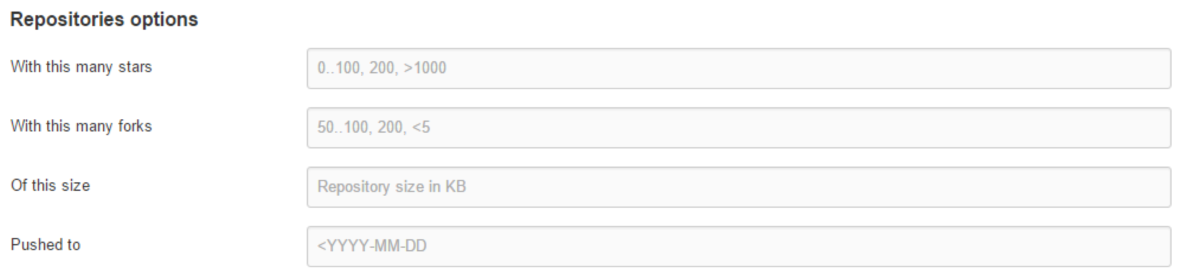
For example, if we want a dialog component written in JavaScript since December 2015, we might use the following search string

```
dialog created:>2015-12-15 language:JavaScript
```

You can use the search option screens to build the syntax dynamically, or you can simply type in the syntax and get the same results.

Repositories options

Similarly, you can restrict the repositories you are interested in searching.



Repositories options	
With this many stars	0..100, 200, >1000
With this many forks	50..100, 200, <5
Of this size	Repository size in KB
Pushed to	<YYYY-MM-DD

Figure 23: Repository search options

You can limit the search to minimum or maximum number of stars and/or forks. You can also restrict the size of the repositories and the date of the latest push.

For example, to find a JavaScript Timezone library used by lots of people, we might use the following syntax:

```
timezone language:javascript stars:>500
```

By combining options, you can narrow down the results returned to just solutions for your application needs.

Tip: Although you can create some pretty large search requests (such as all projects with over 500 stars), you should generally include at least one search word in addition to advanced search options.

As you get comfortable with the search options, you can enter the search text directly without using the advanced search parameter screens.

Tip: Remember, the keywords (stars, language, etc.) all end with a : (colon). Searching for "stars>500" will return no results, but "stars: >500" will work.

Search text options

The following options are available to search the repositories via the search text.

Numeric searching

The **`size:`** and **`forks:`** and **`stars:`** keywords are all numeric based, so you can search using basic comparison operators.

Keyword: numeric

This option finds the exact match (size, number of stars, or forks). It isn't likely you'll know the exact amount for any of the keywords, but it will work for you.

Keyword: >= numeric

This searches for any value greater or equal to the indicated values (use just **>** to not include the numeric value). For example:

```
stars:>500
```

This syntax returns just those repositories that have more than 500 stars.

Keyword: <= numeric

Conversely, the **<=** (or **<**) operator restricts how large of a value to return. If you are looking for project with only a few forks (suggesting a somewhat stable project), you might use the following:

```
forks: <10
```

Keyword: low..high

This finds matches between the specified low and high range (including the boundaries). For example:

```
size: 75000..150000
```

This syntax will return repositories between 75 and 150 MB in size.

Scope searching

The scope operator `IN` lets you search for text in the repository name, its description, or in the readme.md file (or any combination of the three). The general syntax is:

```
<search target> in: name, description, readme
```

For example, I might use the following to search for the **readme.md** files to find any JavaScript code that describes dialogs:

```
dialog in:readme language:javascript
```

To search both `name` and `readme,` include both after the `in:` operator, separated by a comma.

Date searching

When searching repositories, you can use the keyword `created:` to search the creation date and `pushed:` to search the last updated date. You can use it with the greater than, less than, or range operators. For example, to find repositories containing the text `Bootstrap` that were updated after December of 2015, you could use the following:

```
Bootstrap pushed:>2015-12-01
```

Similarly, to find all CSS repositories created during 2015, you could use the following:

```
css created:2015-01-01..2015-12-31
```

Summary

Multiple operators can be combined in one search; only repositories matching all conditions will be returned.

Chapter 6 Other Searches

Although we have focused on repository searches so far, you can also search GitHub for other information. When GitHub searches, it also searches the code, the library, and users at the same time. When I search for Excel in the language of C#, I find over 1,100 repositories and over 150,000 lines of code that contain the word "Excel." In addition, there are over 14,000 issues (bug reports, enhancement requests, etc.) and nine users with "Excel" in their user names.

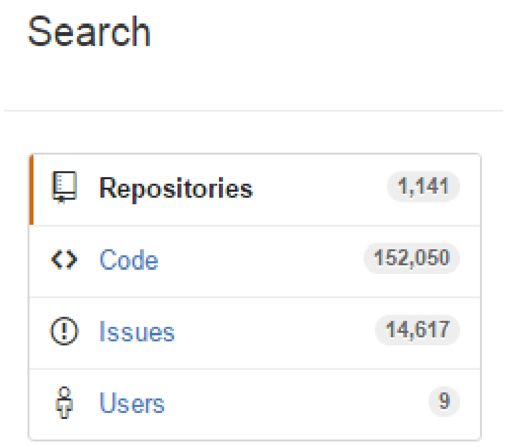


Figure 24: Search options

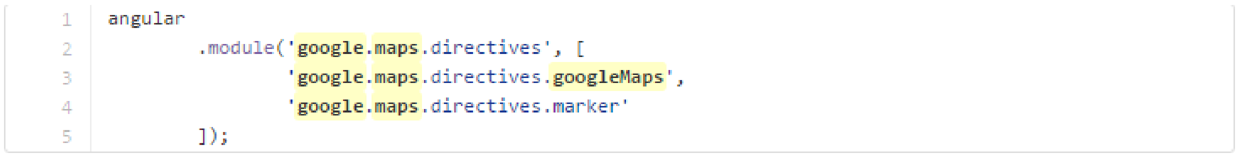
No matter what you're searching for, GitHub shows you the counts and results for the other categories as well. Some of the sort options will vary depending on which category you're looking at. However, the search options described previously can still be applied.

Code search

The code search allows you to search all of the code in the repository looking for strings. You can apply similar rules, only searching for particular dates, languages, etc. The search will look at all matching code in the default branch of the repositories.

Note: GitHub does not let you use wildcards or regular expression when searching code. You will likely get more hits than expected if you search for multiple words.

The code search screen returns every file found in the default branch of a repository that contains the keyword you search for. The matching text will be highlighted when the file name is shown in the search result.



```
angular
        .module('google.maps.directives', [
                'google.maps.directives.googleMaps',
                'google.maps.directives.marker'
        ]);
```

Figure 25: Code search

Tip: One caveat to be aware of for your own repositories: If your code uses an API key of any kind, and you code it directly into your JavaScript, a person searching GitHub could easily come across your key and start using it.

In addition to the options covered in Advanced Searching, there are also a few more code search options.

Search fields (used with IN: operator)

You can search in the file contents (`file`), the path name (`path`), or both together (separated by a comma). You can also use the keyword `extension:` to limit your search to a particular file extension. For example, the old language Clipper used to have a feature called TBROWSE.

```
tbrowse in:path,file extension:prg
```

This search would find all `.PRG` (Clipper file extension) files that use the `TBROWSE` function.

Language search

You can restrict the search to only code written in a particular programming language using the `language:` keyword. For example, to find all JavaScript files using Google Maps, you could use:

```
GoogleMap.js language:JavaScript
```

This search would find all JavaScript code module that include GoogleMap.js, suggesting the code is using Google Maps API calls.

File name search

You can search for a file name using the `filename:` keyword or by file extension using the `extension:` keyword. For example, the following will find all Clipper source files (PRG) using the Class(y) library:

```
extension:prg classy
```

Note: Class(y) was a third-party library that introduced object-oriented classes to Clipper developers.

Sorting the results

The results of the code search can be sorted by best match (the default) or by the date when code was last indexed (GitHub indexes all code in the repository much like Google indexes websites).

Issues

Issues are suggestions, bug reports, documentation updates, etc. that people write about a particular repository. GitHub has very powerful code tracking and comparing capabilities, since a big part of GitHub is collaborating on updates to the projects. There are a number of keywords that allow you to search the issues, perhaps finding some in interesting repositories you might have the experience to help with.

The issue entry is very simple, making it easy to provide an issue and description (and you can attach files). The New Issue screen looks like the following:

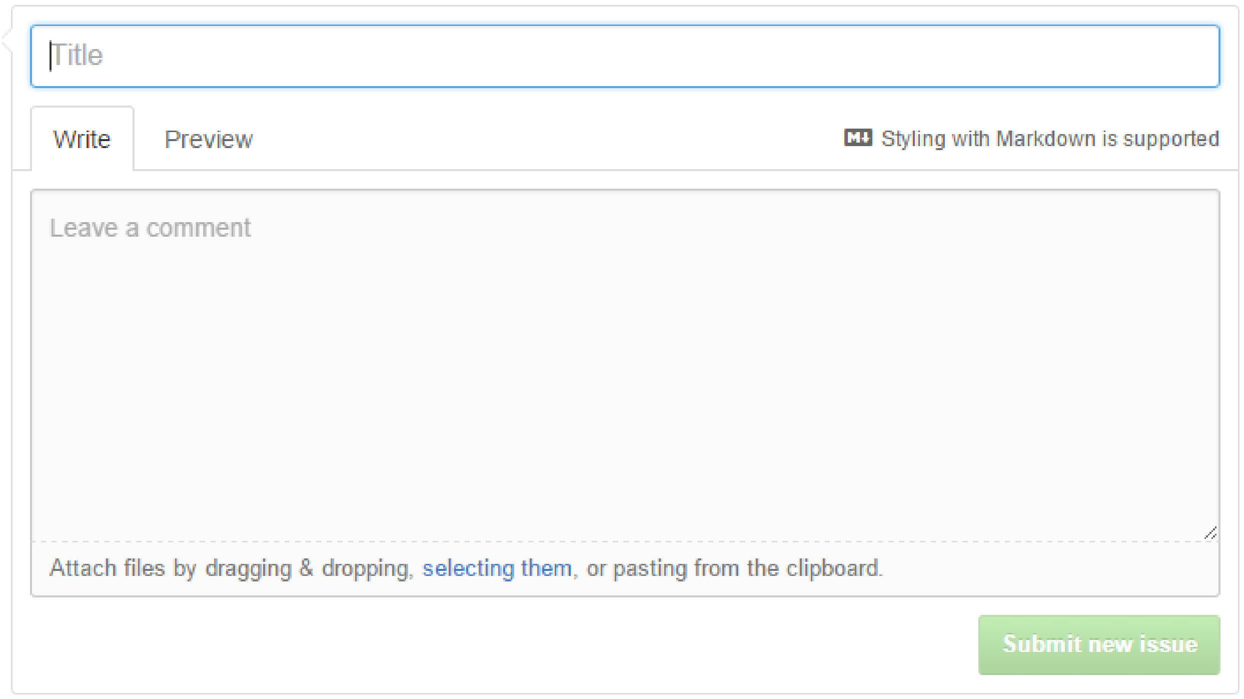

Title
Write
Preview
Styling with Markdown is supported
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Submit new issue

Figure 26: New Issue screen

Before you submit a new issue, you should check to see if similar issues have already been suggested.

Searching the issue content

You can use the **`in:`** keyword with title, body, or comments (or any combination of the three). If you don't specify, then only title and body are searched. For example, imagine you are working with a C# spreadsheet library and are getting corrupted spreadsheets. The following syntax will find all issues with corrupted spreadsheets in C# repositories.

```
corrupt spreadsheet in:title,body language:c#
```

Type of issue

Issues can be issues or pull requests (requests to merge proposed code back into repository). You can limit your search using the **`type:`** keyword:

- **`type:pr`** – Only search pull requests
- **`type:issue`** – Only search issues

If you don't specify a type, both issues and pull requests will be searched.

Users

There are a number of users associated with issues: the original author, the person assigned to fix the issue, commenters, etc. You can use various keywords to search for people involved in various ways with the issue:

- **`Author`** – The person creating the issue
- **`Assignee`** – The user responsible for fixing the issue
- **`Mentions`** – A particular user who might be mentioned in the issue body or comments
- **`Commenter`** – Someone who has commented on an issue

You can combine any combination of the above list when searching. For example, the following search brings up an old issue created by English Extra and commented by Doug Crockford:

```
author:englishextra commenter:douglascrockford
```

Note that issue and corresponding comments can sometimes get a bit testy (read the issue from this search, for example).

If you want to find a user associated with an issue, but aren't sure which role they've had, you can use the **`involves:`** keyword, which is basically just an OR of all four types of users.

Open/Closed

The **state:** keyword can be used to find open versus closed issues. For example, to find open issues related to CSS by Derek Nutile (font-awesome fame):

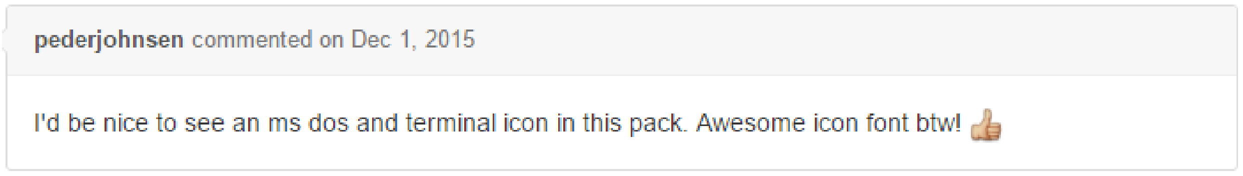
```
state:open involves:dereknutile language:css
```

Dates

You can search when issues were created or updated using the **`created:`** keyword or the **`updated:`** keyword. Both use a date in the format yyyy-mm-dd. For example:

```
font-awesome created:2015-12-01..2015-12-31 language:css
```

This search finds all font-awesome issues that were updated during December of 2015. Font Awesome is an icon library, and often people request specific icons be developed. For example, this user recently suggested an MS-DOS and terminal icon being added to Font Awesome:



pederjohnsen commented on Dec 1, 2015

I'd be nice to see an ms dos and terminal icon in this pack. Awesome icon font btw! 👍

Figure 27: Font Awesome request

Comments

You can use the **`comments:`** keyword to search by the number of comments as issue has attached to it. You can use the standard numeric comparison operators when searching for the comment count.

Sorting issues

You can sort the list of found issues in a number of ways. The default, best match, tries to find the issue that matches most of your search criteria. You can also sort by:

- Most or least commented
- Newest or oldest
- Recently updated or least recently updated

Users

Searching for users or organizations within GitHub adds a few more keywords to help finding repositories. You can combine these keywords to find just the user or company you are searching for.

type:

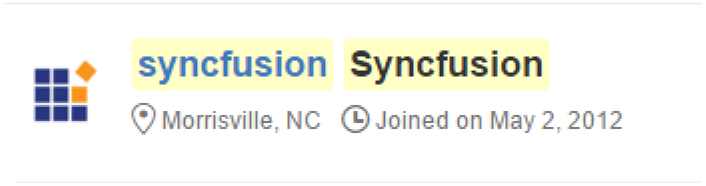
The **`type:`** keyword can be either **`org`** or **`user`**, to find either an organization or a user account. If not specified, GitHub will search both. For example, you might like Erik Mueller's work on the IBM Watson program, so you can search for Erik using the following syntax:

```
erik mueller type:user
```

And to find Syncfusion, you could use the following search:

```
syncfusion type:org
```

The user window will show:



syncfusion Syncfusion

Morrisville, NC Joined on May 2, 2012

Figure 28: SyncFusion organization

in: keyword

You can use the **`in:`** keyword to look at the following fields from the user or organization account:

- **`email`** – Search text appears in email address
- **`login`** – Search text in user's login name
- **`fullname`** – Looks in user's real name

When searching in the full name field, spaces are considered (while in other searches, words separated by spaces are treated as separate words). So the following search will find myself in GitHub users:

```
joe booth in:fullname
```

This makes sense for naming searching, since there are 300+ users with the word "Booth" in their full name, but only two of us with "Joe Booth."

Tip: Another caveat to be aware of: When searching email, the domain portion (@gmail.com) is not considered for the search, so finding all GitHub users with Gmail accounts will not work

Numeric searches

With users, you can use numeric operators (<, =, >, etc.) on the following keywords:

- **`repos`** – The number of repositories the user has
- **`followers`** – The number of people following this user

Location

You can also search for the user's location using the **`location:`** keyword. For example:

```
location:montana language:css
```

This search will find all the GitHub users in the state of Montana with CSS code in their repositories.

```
location:scotland repos:>75 language:javascript
```

This search will find all users from Scotland with more than 75 repositories using JavaScript.

Date searches

When searching users, you can use the keyword **`created:`** to find when users joined GitHub. The dates use the ISO 8601 format (yyyy-mm-dd) and can be > or < and use range searches. For example, to find all United States users who joined in the first quarter of 2008, we could use the following syntax:

```
created:2008-01-01..2008-03-31 location:USA
```

If you want to see the first GitHub users (GitHub was started in October 2007), you can run the following search:

```
created:2007-10-20
```

Summary

GitHub determines what type of search you are using by the keywords it finds in your search text. It will apply the keyword to all applicable groups (repositories, issues, code, users). For example, the **`created:`** keyword applies to both repositories and users, so GitHub will return both repositories created on the date and users who joined on that date.

Keyword table

The following table shows the various keywords and which searches they apply to:

Table 1: GitHub search keywords

Keyword	Repos	Code	Issues	Users
`assignee`			X	
`author`			X	
`commenter`			X	

Keyword	Repos	Code	Issues	Users
`comments`			X	
`created`	X		X	X
`extension`		X		
`filename`		X		
`followers`				X
`forks`	X			
`in:body`			X	
`in:comment`			X	
`in:description`	X			
`in:email`				X
`in:file`		X		
`in:fullname`				X
`in:login`				X
`in:name`	X			
`in:path`		X		
`in:readme`	X			
`in:title`			X	
`label`				
`language`	X	X	X	X
`location`				X
`mentions`			X	
`pushed`	X			
`repos`				X
`size`	X	X		X

Keyword	Repos	Code	Issues	Users
`stars`	X			
`state`			open, closed	
`type`			pr, issue	org, user
`updated`			X	

Chapter 7 Browsing GitHub

Although searching is a powerful feature of GitHub, you can also explore the many interesting projects and see the latest trends. Under your account icon, pull down the menu and select **Explore**. You will see a screen similar to the following:

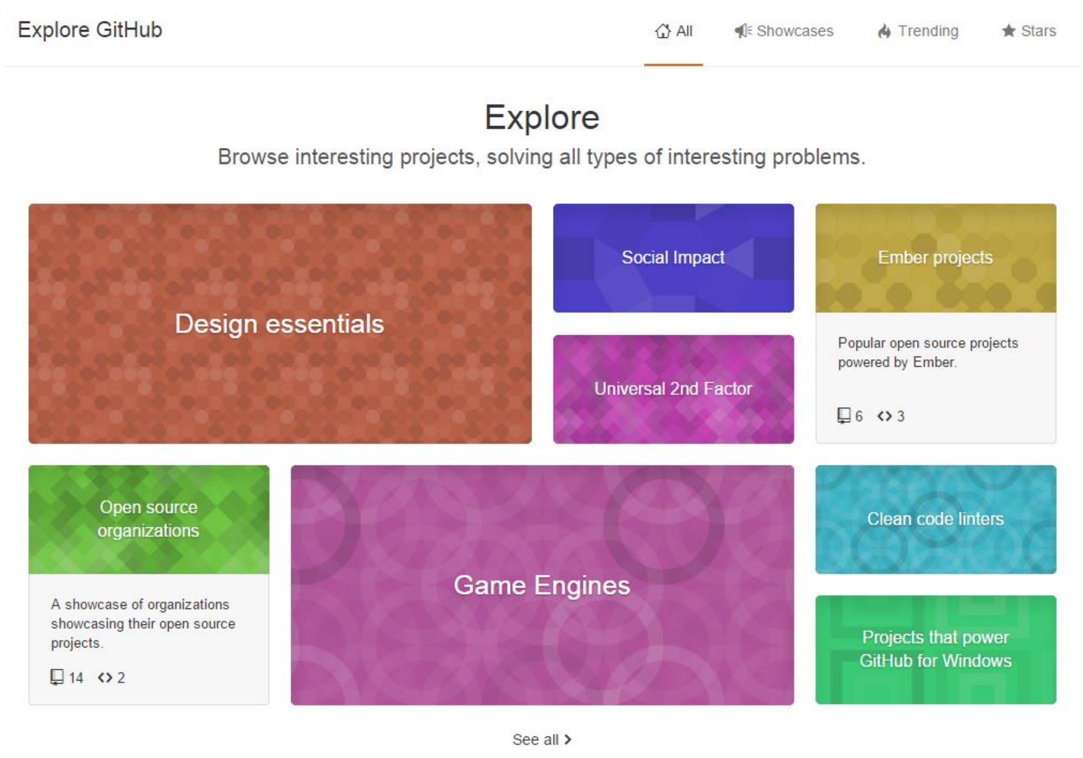
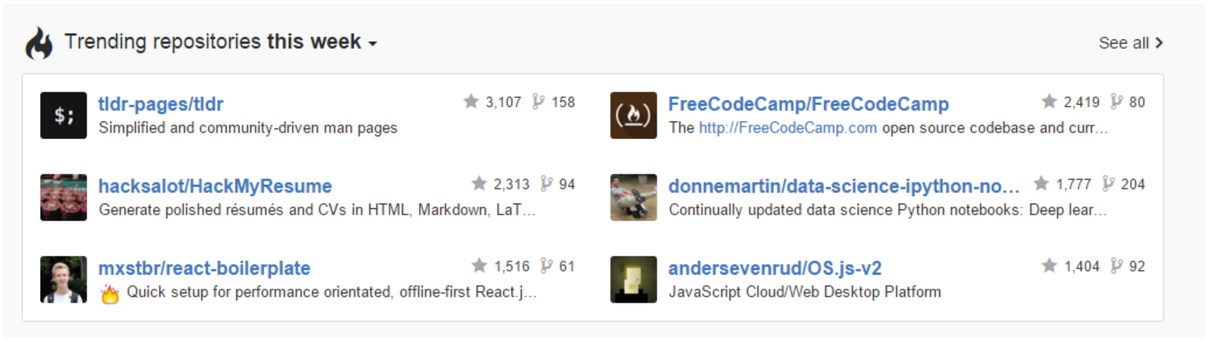

Explore GitHub
All
Showcases
Trending
Stars
Explore
Browse interesting projects, solving all types of interesting problems.
Design essentials
Social Impact
Ember projects
Popular open source projects powered by Ember.
6
3
Universal 2nd Factor
Open source organizations
A showcase of organizations showcasing their open source projects.
14
2
Game Engines
Clean code linters
Projects that power GitHub for Windows
See all

Figure 29: GitHub Explore topics

You can click on any of the boxes to see the most-starred repositories in the various categories. The category list may change depending on the latest trends and activity in GitHub.

Below the categories, you'll find the trending repositories (based on activity for the week).


Trending repositories this week

See all

tldr-pages/tldr ★ 3,107 158
Simplified and community-driven man pages

FreeCodeCamp/FreeCodeCamp ★ 2,419 80
The http://FreeCodeCamp.com open source codebase and curr...

hacksalot/HackMyResume ★ 2,313 94
Generate polished résumés and CVs in HTML, Markdown, LaT...

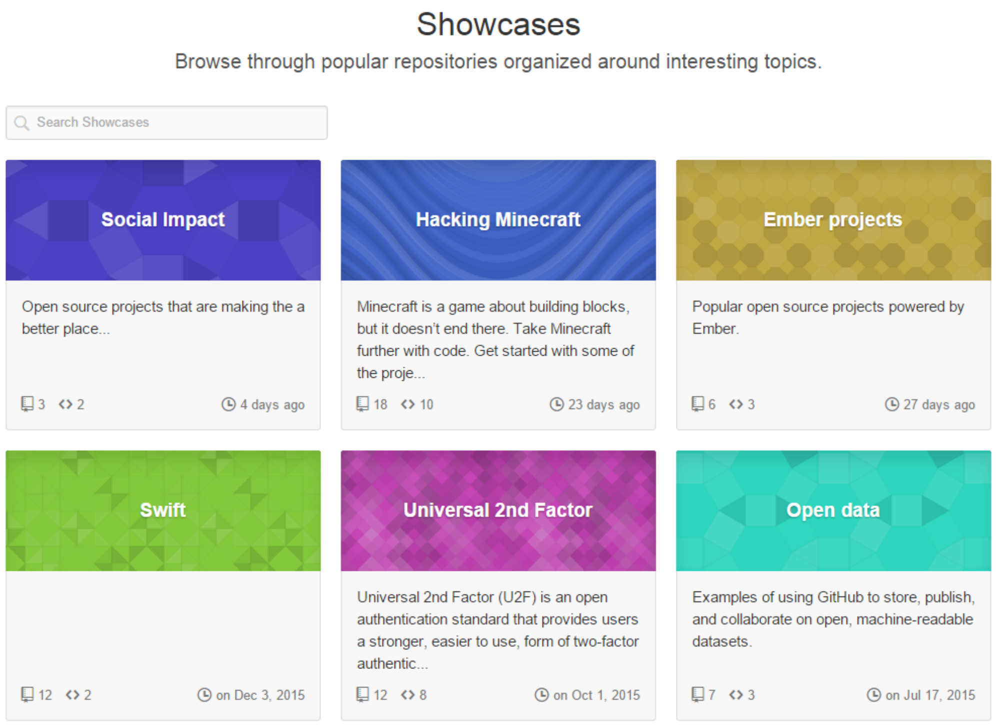
donnemartin/data-science-ipython-no... ★ 1,777 204
Continually updated data science Python notebooks: Deep lear...

mxstbr/react-boilerplate ★ 1,516 61
Quick setup for performance orientated, offline-first React.j...

andersevenrud/OS.js-v2 ★ 1,404 92
JavaScript Cloud/Web Desktop Platform

Figure 30: Trending repositories

The next menu option, Showcases, shows various topics of interest and lets you explore the various repositories within them.



Showcases

Browse through popular repositories organized around interesting topics.

Social Impact

Open source projects that are making the a better place...

3 2 4 days ago

Hacking Minecraft

Minecraft is a game about building blocks, but it doesn't end there. Take Minecraft further with code. Get started with some of the proje...

18 10 23 days ago

Ember projects

Popular open source projects powered by Ember.

6 3 27 days ago

Swift

12 2 on Dec 3, 2015

Universal 2nd Factor

Universal 2nd Factor (U2F) is an open authentication standard that provides users a stronger, easier to use, form of two-factor authentic...

12 8 on Oct 1, 2015

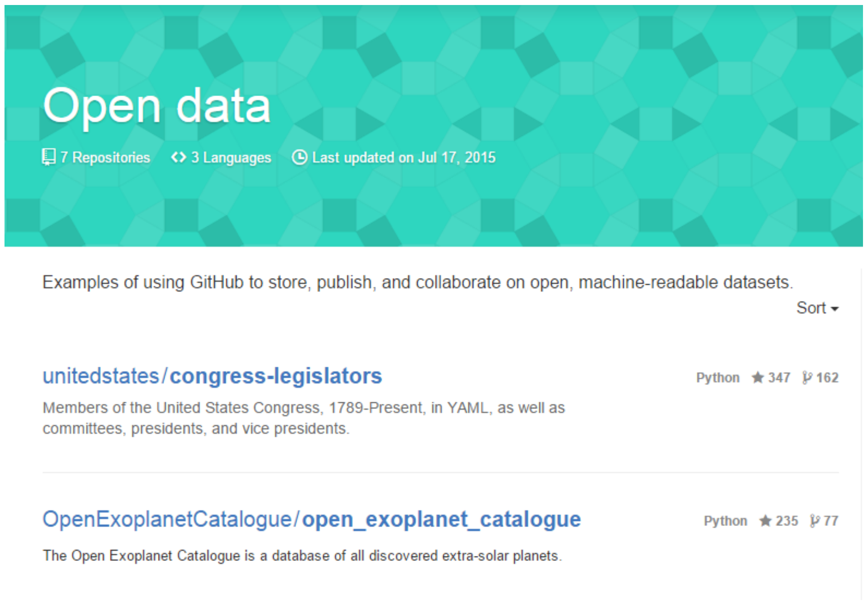
Open data

Examples of using GitHub to store, publish, and collaborate on open, machine-readable datasets.

7 3 on Jul 17, 2015

Figure 31: Showcases

You can click on any box to see the repositories within. For example, the following screen appears when you choose the Open data box.



Open data

7 Repositories 3 Languages Last updated on Jul 17, 2015

Examples of using GitHub to store, publish, and collaborate on open, machine-readable datasets.

unitedstates/**congress-legislators** Python ★ 347 162

Members of the United States Congress, 1789-Present, in YAML, as well as committees, presidents, and vice presidents.

OpenExoplanetCatalogue/**open_exoplanet_catalogue** Python ★ 235 77

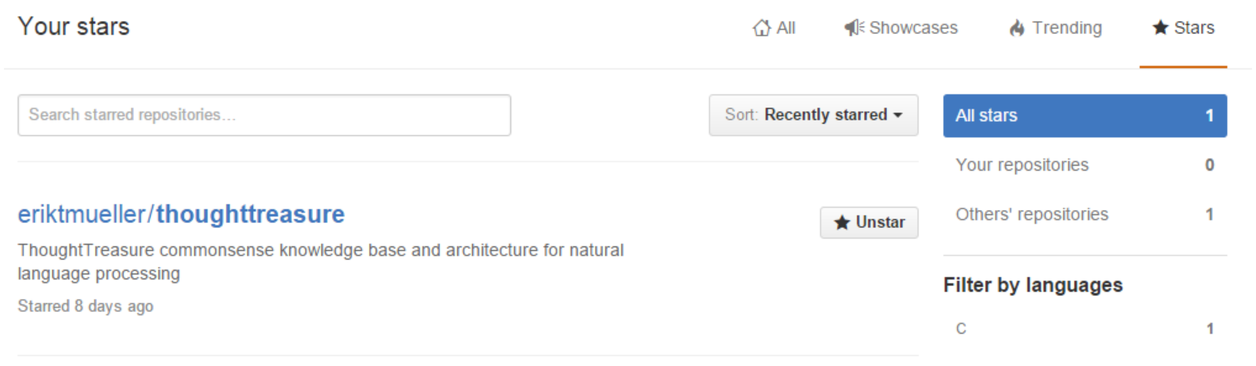
The Open Exoplanet Catalogue is a database of all discovered extra-solar planets.

Figure 32: Open data repositories

Sometimes, you might just want to explore GitHub and see what other developers are actively working on. Or you might not quite know what the search words are, but seeing the categories can help you find a useful repository. The Showcases and Trending repositories can be a nice alternative to searching by keywords.

Stars

The Stars menu option will show a list of any repositories that you have starred:



eriktmueller/thoughttreasure

ThoughtTreasure commonsense knowledge base and architecture for natural language processing

Starred 8 days ago

Figure 33: Starred repositories

Summary

Exploring GitHub can be a fun exercise, and you never know what goodies you might find, including those you probably never even thought to search for.

Chapter 8 Issues – In Detail

Issues are the collaboration component of GitHub, where people communicate about project activity. While Git and GitHub are powerful version control tools, the collaboration support available through issues and pull requests encourage people to work together to produce great software projects.

Creating an issue

From the project detail page, you can click on the **Issues** tab and click on the **New issue** button.

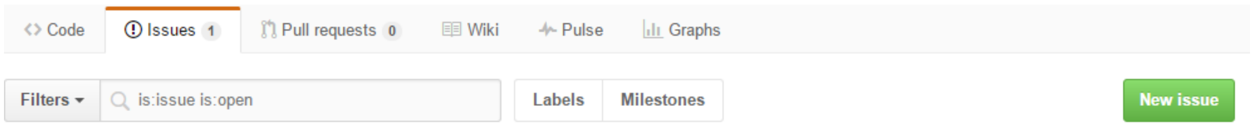


Figure 34: Issues tab

When you click on the New issue button, the following screen will appear to let you enter your issue details. You can use the Markdown syntax (see Appendix 1) to style your text.

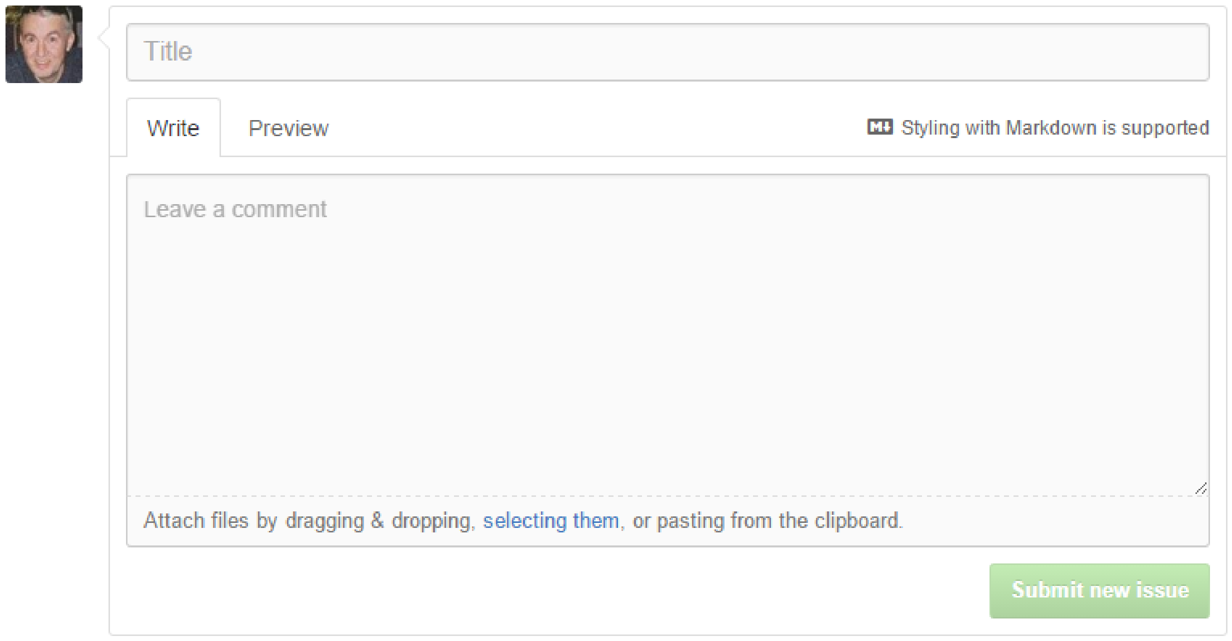
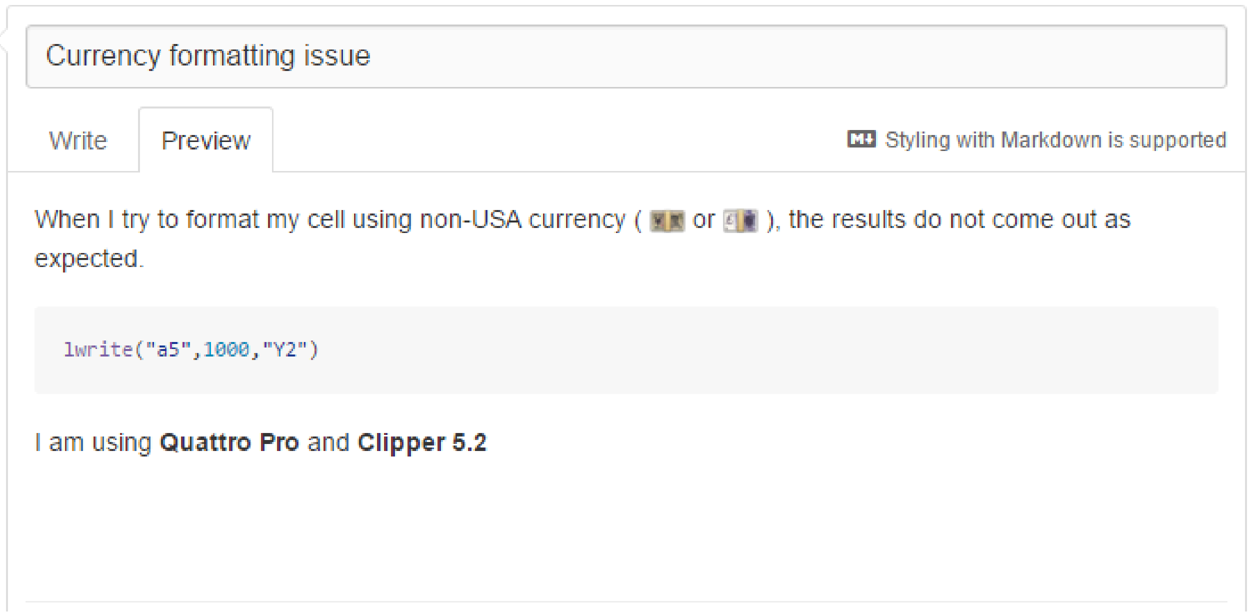

Title
Write
Preview
Styling with Markdown is supported
Leave a comment
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Submit new issue

Figure 35: Writing an issue

You can use both Markdown and emoji in your body text to format the issue description. The Preview tab shows you what the issue will look like.



Currency formatting issue

When I try to format my cell using non-USA currency (or), the results do not come out as expected.

```
lwrite("a5",1000,"Y2")
```

I am using **Quattro Pro** and **Clipper 5.2**

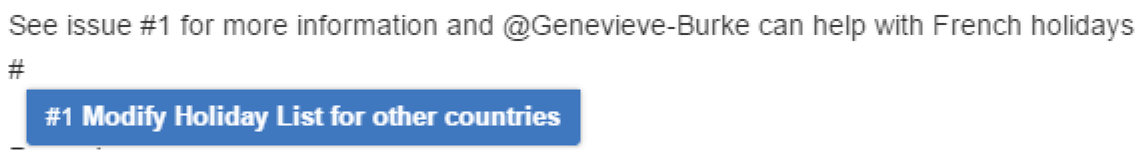
Figure 36: Issue preview

You can also attach files to the issue to help the repository author understand the issue clearly. Anyone watching the repository will now receive an email with the issue content.

Tip: Keep in mind that GitHub is international—be sure to express your issue as clearly as possible, since people from all over the world may be reviewing the issue.

In addition to Markdown, you can also use the **`@userName`** syntax to refer to a GitHub user and the **`#Issue`** syntax to reference another issue or pull request. This allows issues to be cross-referenced and to be linked with other user and issues.

When you are typing an issue, GitHub recognizes the **`@`**, **`#`**, and **`:`** characters, and will search for matches to make it easier. For example, typing the **`#`** brings up a list of other issues within the repository:



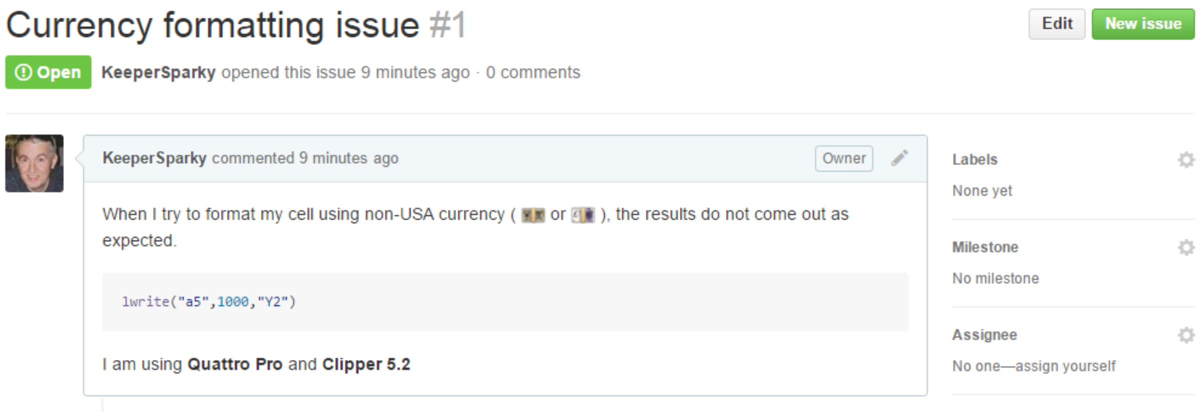
See issue #1 for more information and @Genevieve-Burke can help with French holidays
#

Figure 37: List of issues within edit window

The **`:`** (colon) will bring up a list of support emoji characters, and the **`@`** symbol will bring up a list of users associated with the repository and issues.

Editing the issue

When the repository owner sees the issue, they will be able to start working with the issue. This allows them to categorize the issue, assign it to someone, marked it closed, and to comment about the issue.


Currency formatting issue #1
Edit
New issue
Open
KeeperSparky opened this issue 9 minutes ago · 0 comments
KeeperSparky commented 9 minutes ago
Owner
Labels
None yet
Milestone
No milestone
Assignee
No one—assign yourself

When I try to format my cell using non-USA currency (or), the results do not come out as expected.

```
lwrite("a5",1000,"Y2")
```

I am using **Quattro Pro** and **Clipper 5.2**

Figure 38: Owner view

Labeling the issue

Generally, the person reviewing the issue will first assign a label to it, such as bug or enhancement.

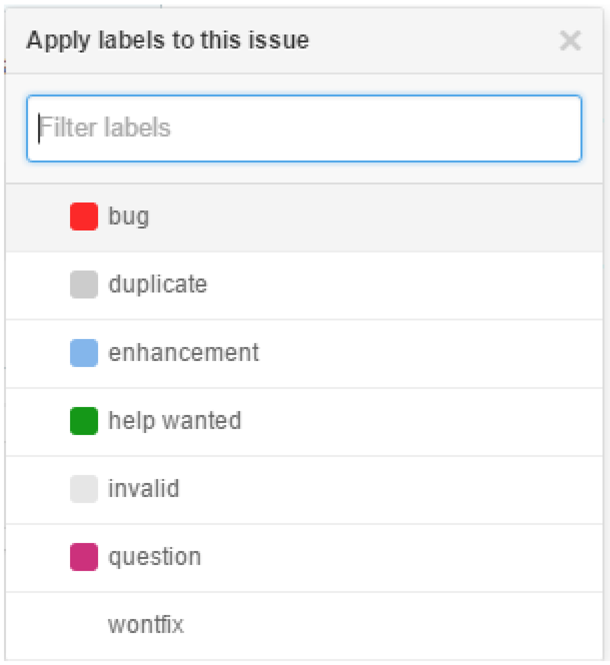


Figure 39: Labeling the issue

The seven items shown in Figure 39 are the default labels that GitHub offers. You can add new labels from the issue list by clicking on the **Labels** button.

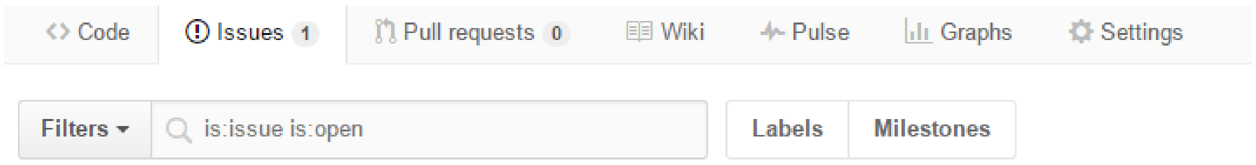


Figure 40: Issues list

This will display a screen that allows you to edit labels, remove ones you don't want, and add any new ones. When you add a new label, GitHub asks for the label name and has a color picker to let you choose a color to use for this issue.

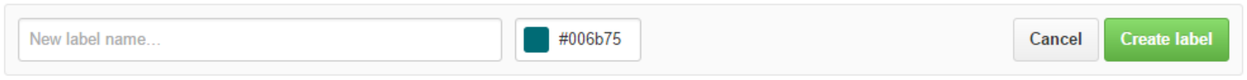


Figure 41: New issue

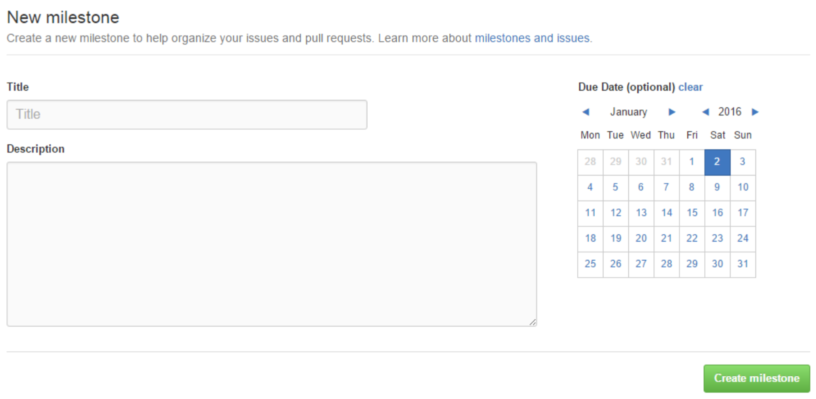
Once you've added the new label, you can use that label for all new and existing issues.

Milestones

If you have a number of issues, you might want to group them into milestones. A milestone can be anything you want, such as:

- Sprint – Issues to work on in the next two weeks
- Beta release – Work for next major release
- Customer specific – Issues for a particular customer

You can create milestones similarly to how you create labels. The milestone will also allow you an optional due date to help with tracking.



New milestone

Create a new milestone to help organize your issues and pull requests. Learn more about milestones and issues.

Figure 42: Milestone screen

Assignee

The Assignee button lets you assign a person to work on this issue. It should be the owner or any collaborator. You can add a collaborator from the settings menu when you (the owner) are viewing the repository.

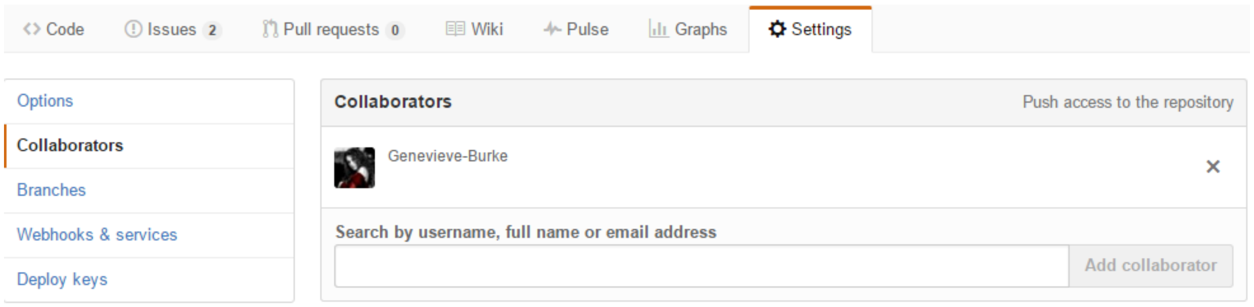
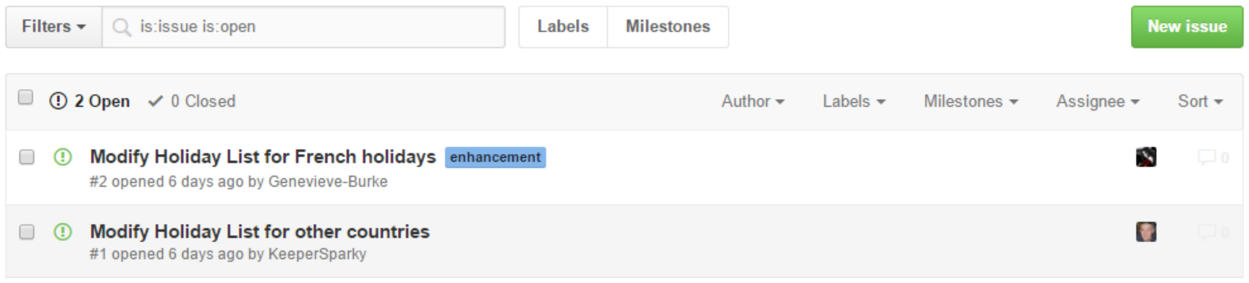


Figure 43: Collaborator

The collaborator can have tasks assigned to them. You can see the list of collaborators and add a new one by looking up user name, email address, or full name. You can use the X button to remove a collaborator from the project. A collaborator will also have push access to your project as well.

Viewings issues

As you work with issues, adding labels, assignees, etc., GitHub will display all of the information in the issue list



Filters ▾ | is:issue is:open | Labels | Milestones | New issue

2 Open ✓ 0 Closed — Author ▾ Labels ▾ Milestones ▾ Assignee ▾ Sort ▾

Modify Holiday List for French holidays enhancement
#2 opened 6 days ago by Genevieve-Burke

Modify Holiday List for other countries
#1 opened 6 days ago by KeeperSparky

Figure 44: Issue list

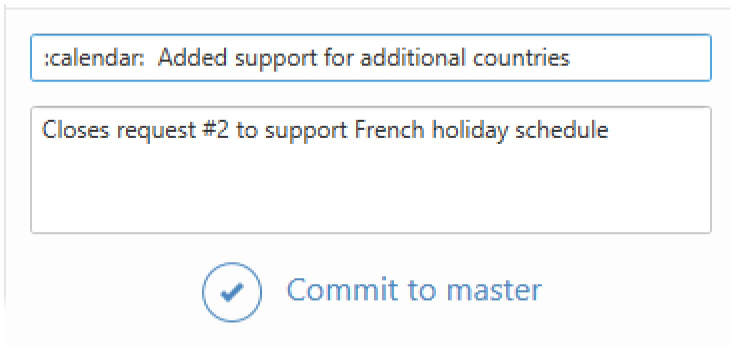
You can use the various options across the top (author, labels, milestones, and assignee) to filter the issue list. You can also sort the list the using same options you can use while searching. GitHub makes tracking issues both simple and powerful.

Closing issues

When you are making a commit to the repository, you can close the issue via the commit message. GitHub recognizes the following words as meaning the issue is being closed. You should follow the keyword with the #issue number.

- **`Close`**
- **`Closes`**
- **`Closed`**
- **`Fix`**
- **`Fixes`**
- **`Fixed`**
- **`Resolve`**
- **`Resolves`**
- **`Resolved`**

When GitHub sees the pattern in the commit message, it will automatically close the issue. You can close multiple issues in one commit message as well. For example, the following commit message will automatically close issue #2.


:calendar: Added support for additional countries

Closes request #2 to support French holiday schedule

Figure 45: Closing an issue in commit message

Summary

The issue tracking and collaboration features of GitHub make it very useful as a tool to track development progress and work. It is a simple, yet powerful system to learn, and is very useful for management of projects with multiple people, multiple milestones, etc.

Chapter 9 Viewing a Project

When the results of a search are displayed, there's a lot of information included in the repository list. This information can give you an idea of how well received the project is and how actively it's being worked on.

Search Results

The following example is from a search of JavaScript code that works with the HTML Canvas. **`three.js`** is a JavaScript library that lets you do 3D drawing and animation on the HTML 5 Canvas. It's designed to be simple to use, yet is a pretty powerful library for 3D work.

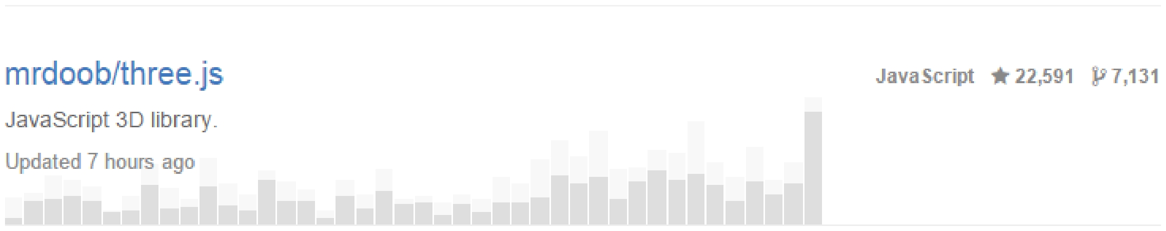


Figure 46: Search results

A look at the search results shows some information that might help get a sense of the project.

Repository Name

In Figure 46, the **`mrdoob/three.js`** is the owner and name of the repository. If you open the repository window, you will be able to click on the owner's name to find out more about him or her, and possibly other repositories they might have. For example, opening one of my projects (**`KeeperSparky`**) will show you the two projects I've uploaded while working on this book.

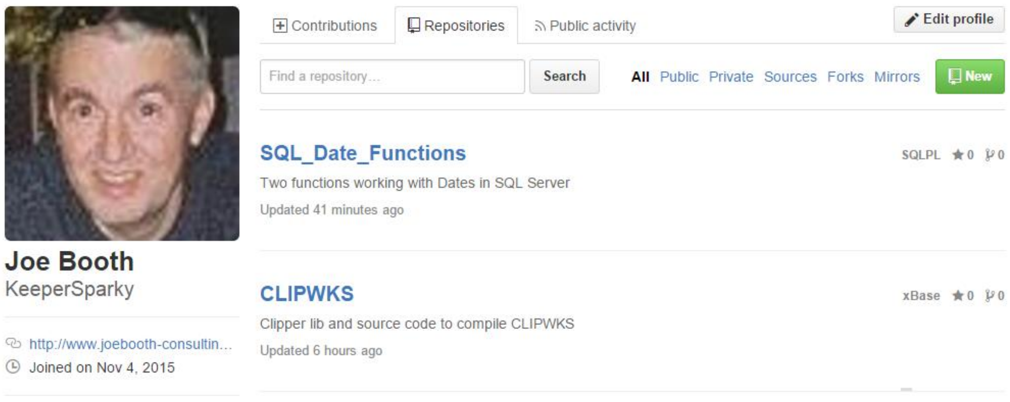


Figure 47: Owner profile

In case you're wondering, it's "KeeperSparky" because I play soccer goalie and had the nickname Sparky (Radar's friend from M.A.S.H.) during my college days.

Language

In the right-hand top corner, you see three pieces of information: the language the project is written in, the number of stars (positive ratings by GitHub users), and the number of forks (people enhancing or working on the project).

Note: Forks is a historical number tracking all forks since the repository was first added (not 7,000 active forks).

Last updated

Under the repository name and description is an indication of when the repository was last updated; in our example, it was updated 7 hours ago. This is generally a good indication that the project is pretty active.

Participation graph

Below the description and last updated information is the participation graph. The graph shows activity in terms of commits to the repository, from both the owner and contributors.

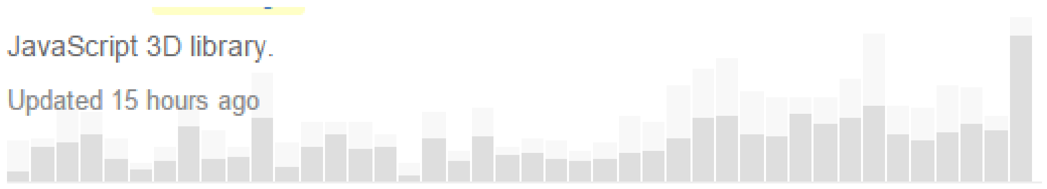


Figure 48: Participation graph

The owner and contributors making the commits are differentiated, so a project like the above shows lot of activity, both from Mr.Doob and the 519 contributors…

Repository details

Once you click on a repository, there's a lot of information presented, and a lot of interaction you (as a registered user) can do with the repository. We will cover the various screens in this section to help guide you to the options you might be interested in.

Social interaction

The bar at the top of the page shows the repository owner and name on the left, and three buttons on the right.

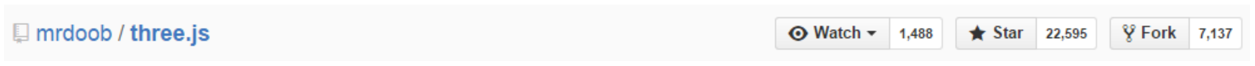


Figure 49: Top bar of repository

Watch

You can watch it (be notified of all activity) it, you can ignore it (never see any notifications), or you can "not watch" (only see notifications specifically mentioning your user ID or that you are explicitly participating in).

Star

The Star is similar to the Like button in Facebook; it generally means you've used the project and like it. You can star or un-star at any time by toggling the Star button. The number to the right shows how many users have "starred" or "liked" the project.

Fork

The Fork option creates a copy of the repository for your user ID (it will appear with your other repositories showing the fork icon instead). A fork is generally a copy you wish to either contribute to or to start your own project based on. You can download a project if you simply want to use the code, without creating a fork.

Repository actions

Just below the repository name is a set of tabs showing various views of the project. In addition, you can see the number of commits (updates over project life), branches, releases, and contributors.

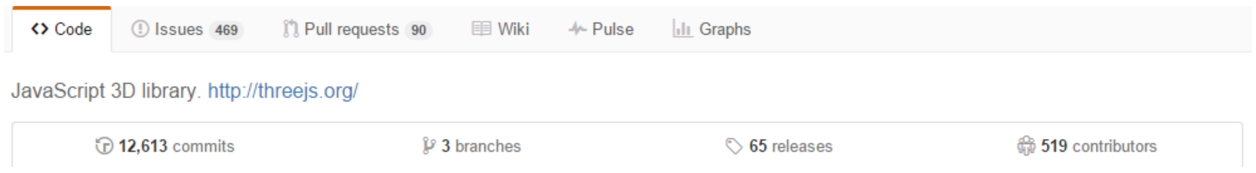
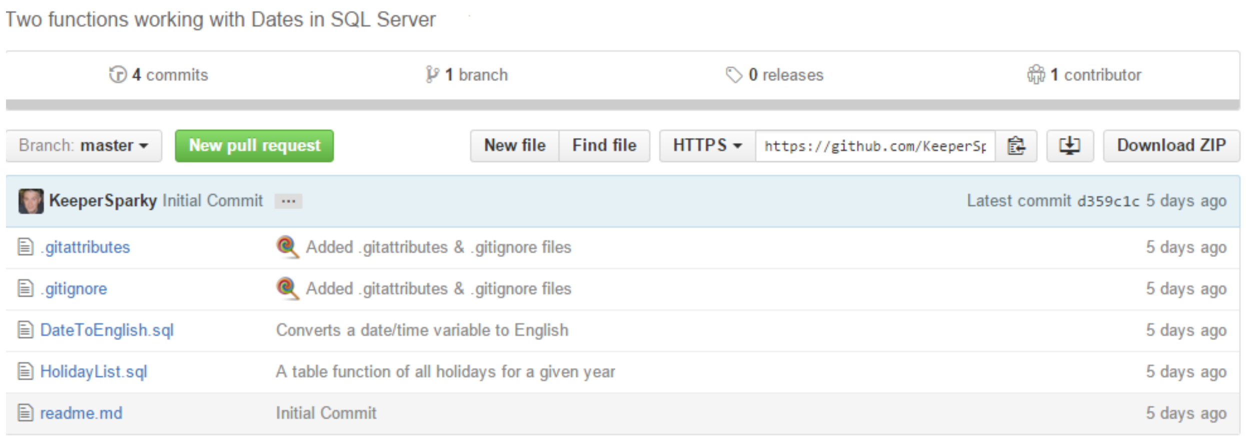


Figure 50: Repository actions

Code tab

The <> Code tab is the default selected view when you find a repository. You can see the actual files and any subfolders that exist in the repository.


Two functions working with Dates in SQL Server

4 commits | 1 branch | 0 releases | 1 contributor

Branch: master | New pull request | New file | Find file | HTTPS | https://github.com/KeeperSp | Download ZIP

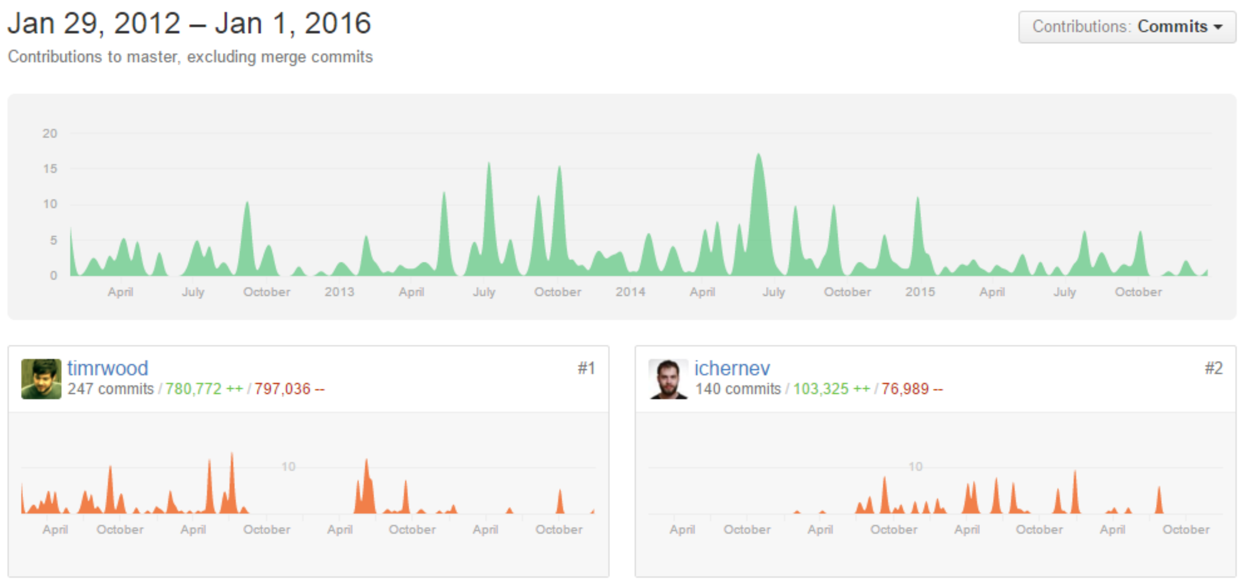
KeeperSparky Initial Commit ··· Latest commit d359c1c 5 days ago

.gitattributes	Added .gitattributes & .gitignore files	5 days ago
.gitignore	Added .gitattributes & .gitignore files	5 days ago
DateToEnglish.sql	Converts a date/time variable to English	5 days ago
HolidayList.sql	A table function of all holidays for a given year	5 days ago
readme.md	Initial Commit	5 days ago

Figure 51: Code tab

The very top line shows the summary line that was added when repository was first created. If you're the owner of the repository, you'll be able to edit this line within GitHub.

The next line shows the activity on the code within the repository (the number of commits, branches, releases, and contributors). You can click on each item to see more details. For example, if we look at **`momentjs`** (a JavaScript time library), we find 90 contributors.


Jan 29, 2012 – Jan 1, 2016

Contributions to master, excluding merge commits

Contributions: Commits

timrwood #1 — 247 commits / 780,772 ++ / 797,036 --

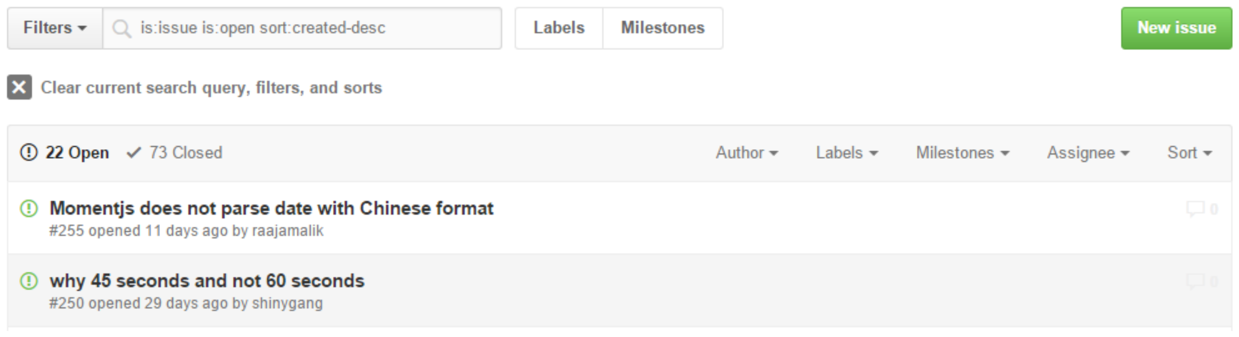
ichernev #2 — 140 commits / 103,325 ++ / 76,989 --

Figure 52: Contributors

The screen shot in Figure 52 only shows the top two contributors (based on number of commits) but GitHub will show all contributors (within reason; if a project has hundreds, GitHub will warn you and not show them all). Next to each person's name is the number of commits, followed by the number of lines added (**green**) and the number of lines deleted (**red**). GitHub offers a lot of number crunching and graphs to explore.

Issues tab

Clicking on the Issues tab brings up a searchable list of issues associated with the repository. You can also create a new issue by clicking on the green New issue button.



Momentjs does not parse date with Chinese format
#255 opened 11 days ago by raajamalik

why 45 seconds and not 60 seconds
#250 opened 29 days ago by shinygang

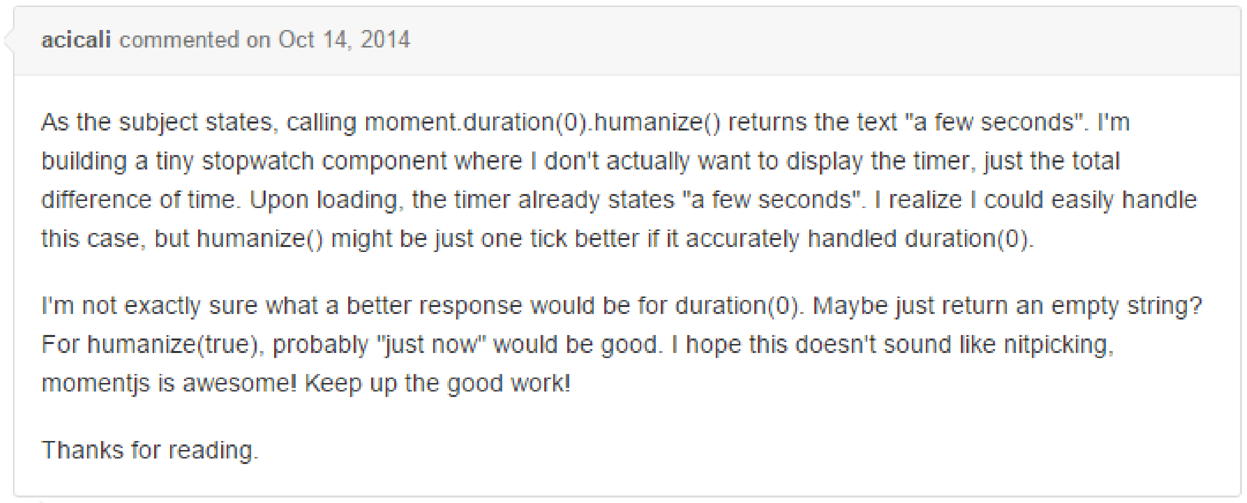
Figure 53: Issues

From this screen, you can filter the issue list by clicking on the drop-down arrows next to **Author** (creator of the issue), **Labels** (such as bug, enhancement, new icon, etc.), **Milestones**, and **Assignee**. You can also sort the list by date and number of comments added to the issue.

The left side allows you to toggle between open and closed issues.

The **Filters** dropdown menu lets you search the issue list for issues you've created, issues that have been assigned to you, issues mentioning you, etc.

When you are writing an issue, be sure to provide enough detail to make it clear what you are reporting. If it's a bug, include steps to reproduce it. If it's a feature request, be sure to make it clear what you want added or enhanced. The following example shows a clear issue written to the **`momentjs`** library.



acicali commented on Oct 14, 2014

As the subject states, calling moment.duration(0).humanize() returns the text "a few seconds". I'm building a tiny stopwatch component where I don't actually want to display the timer, just the total difference of time. Upon loading, the timer already states "a few seconds". I realize I could easily handle this case, but humanize() might be just one tick better if it accurately handled duration(0).

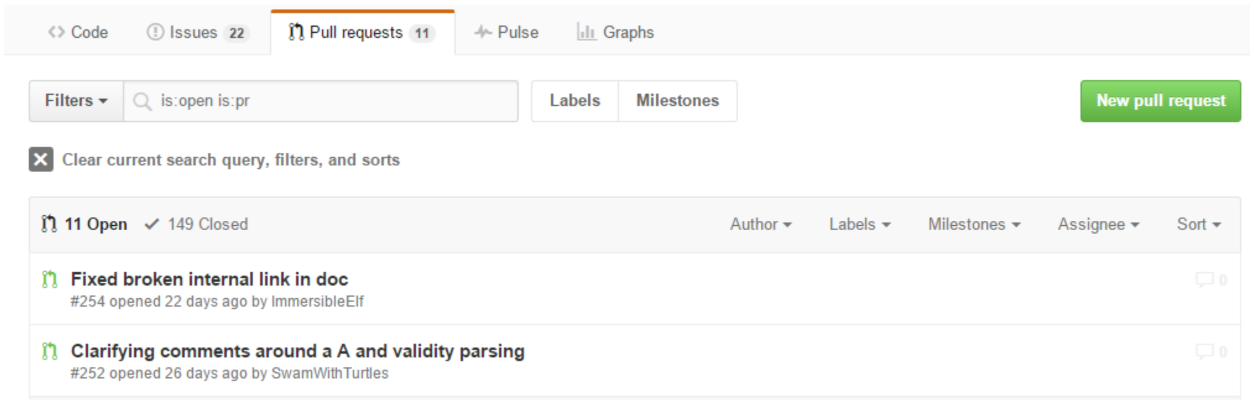
I'm not exactly sure what a better response would be for duration(0). Maybe just return an empty string? For humanize(true), probably "just now" would be good. I hope this doesn't sound like nitpicking, momentjs is awesome! Keep up the good work!

Thanks for reading.

Figure 54: Example issue

Pull requests

A pull request means that a fix has been made in response to an issue, and the collaborator is requesting that the fix be merged back into one of the branches. Figure 55 shows a sample pull request list.



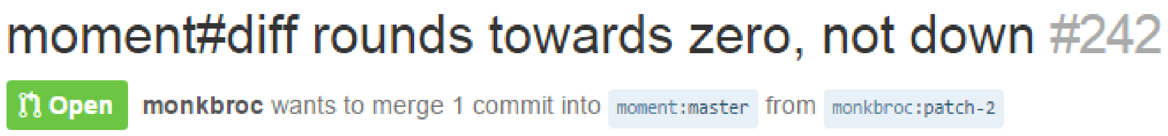
Fixed broken internal link in doc
#254 opened 22 days ago by ImmersibleElf

Clarifying comments around a A and validity parsing
#252 opened 26 days ago by SwamWithTurtles

Figure 55: Pull request list

Pull requests have similar filtering and sorting options to the Issues tab (such as Author, Label, and Milestone).

When you open a pull request, the first line shows the request as a comment.



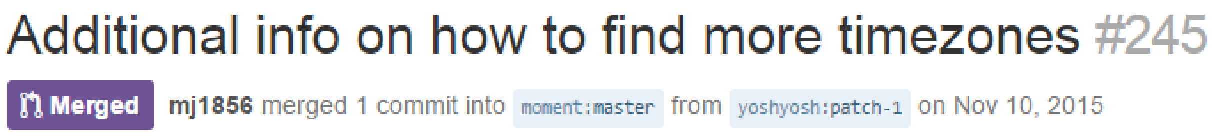
moment#diff rounds towards zero, not down #242

Open monkbroc wants to merge 1 commit into moment:master from monkbroc:patch-2

Figure 56: Pull request description

In this example, the user **`monkbroc`** has updated his forked version (**`monkbroc:patch-2`**) and is requesting that his commit to his fork be merged into the master branch for the project.

You might also see other types of descriptions with the pull request. For example, Figure 57 shows a closed pull request that was merged back into the main branch.

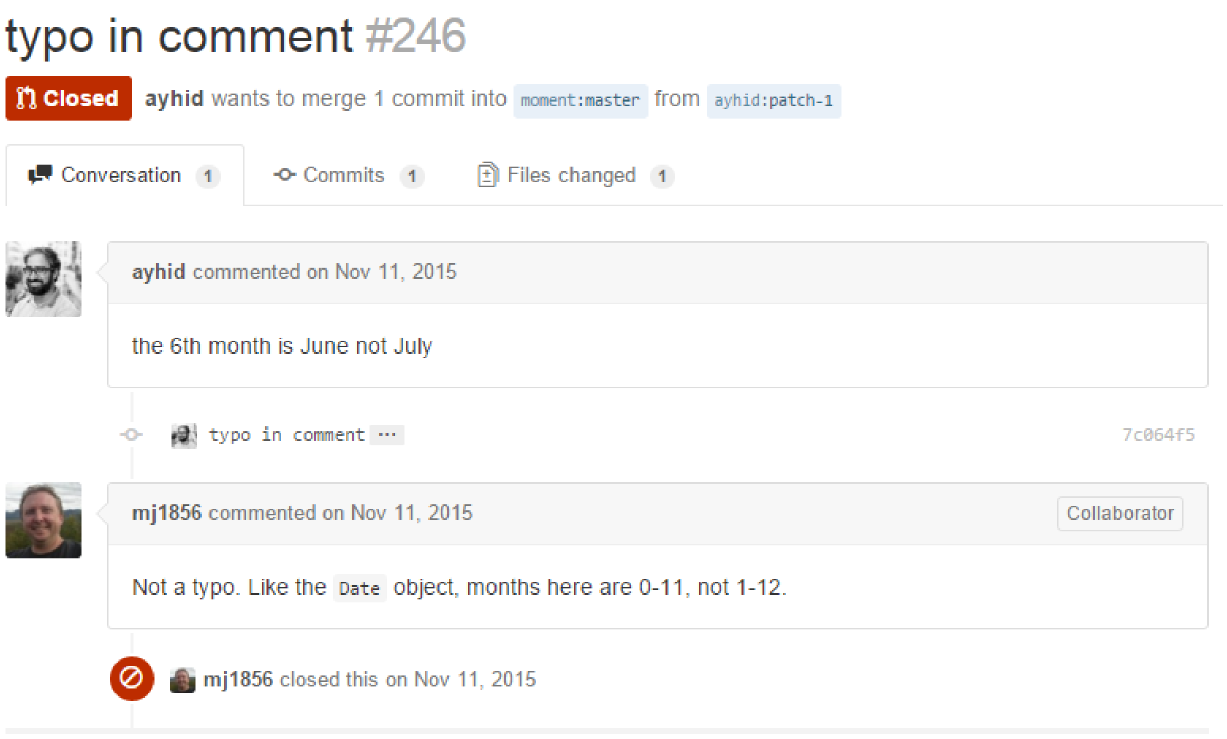


Additional info on how to find more timezones #245

Merged mj1856 merged 1 commit into moment:master from yoshyosh:patch-1 on Nov 10, 2015

Figure 57: Merged pull request

However, not all pull requests get merged into the master branch; in Figure 58, the pull request was closed without being merged. It’s up to the repository owner to accept and apply any pull requests.


typo in comment #246

Closed ayhid wants to merge 1 commit into `moment:master` from `ayhid:patch-1`

Conversation 1 Commits 1 Files changed 1

ayhid commented on Nov 11, 2015

the 6th month is June not July

typo in comment … 7c064f5

mj1856 commented on Nov 11, 2015 Collaborator

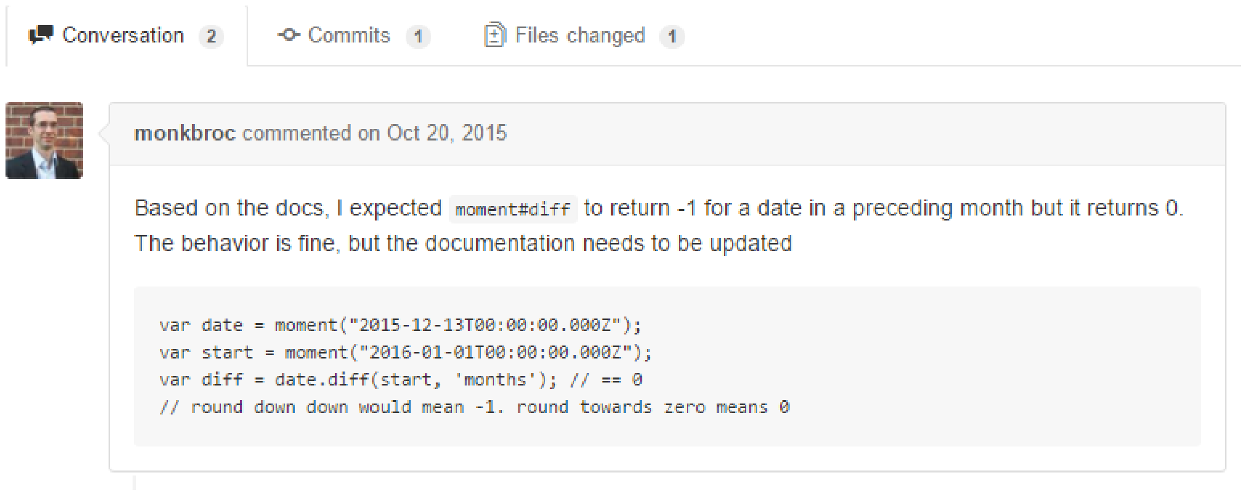
Not a typo. Like the `Date` object, months here are 0-11, not 1-12.

mj1856 closed this on Nov 11, 2015

Figure 58: Example of closed pull request

On the pull request list, the different icons indicate the status of the request.

On the next line, you'll see three tabs. The first tab, Conversation, shows the various issues and comments associated with the request.


Conversation 2 Commits 1 Files changed 1

monkbroc commented on Oct 20, 2015

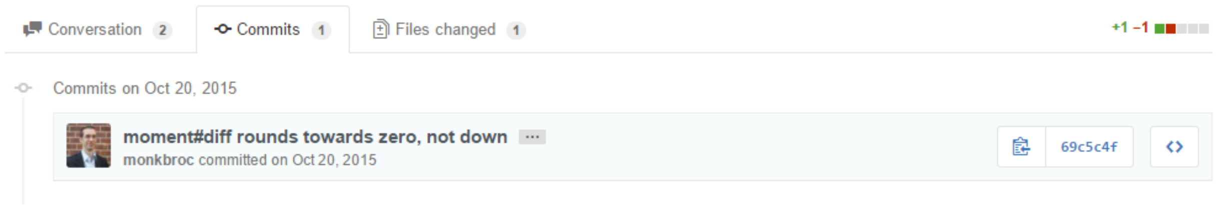
Based on the docs, I expected `moment#diff` to return -1 for a date in a preceding month but it returns 0. The behavior is fine, but the documentation needs to be updated

```
var date = moment("2015-12-13T00:00:00.000Z");
var start = moment("2016-01-01T00:00:00.000Z");
var diff = date.diff(start, 'months'); // == 0
// round down down would mean -1. round towards zero means 0
```

Figure 59: Pull request conversation

This tab allows you to see the original issue and subsequent comments.

The next tab, Commits, shows any commits to their forked branch that have been made by the collaborator. This shows the date and commit message from the fork.



Commits on Oct 20, 2015

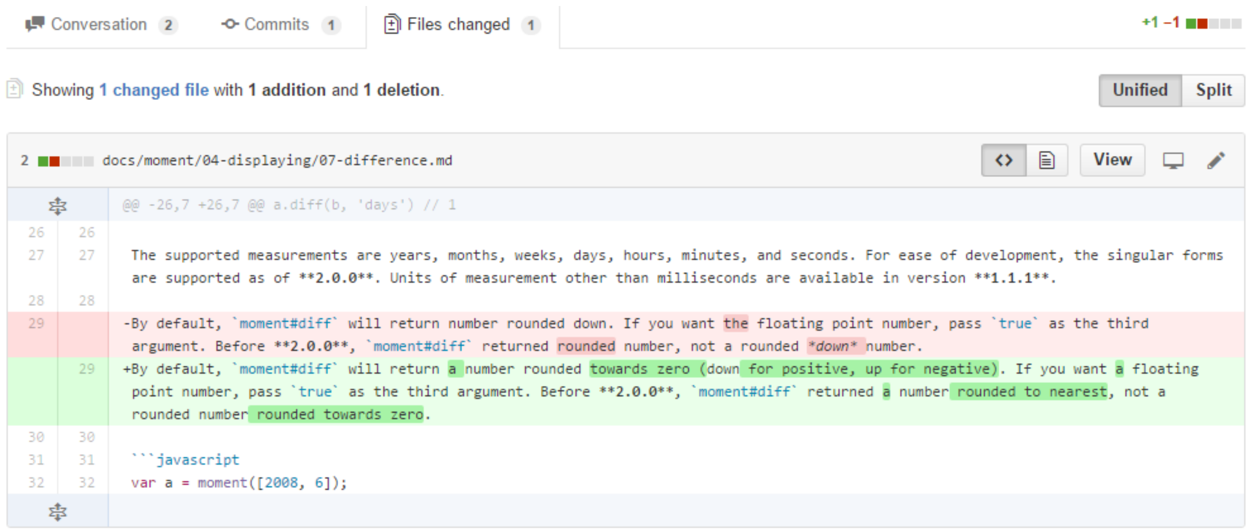
moment#diff rounds towards zero, not down
monkbroc committed on Oct 20, 2015

69c5c4f

Figure 60: Commit message

The color-coded numbers show the number of lines added and removed during this commit.

The final tab, Files changed, shows the details as to exactly what was changed in the file(s) with this commit. This allows the owner to see whether or not she wants to accept the pull request.



Showing 1 changed file with 1 addition and 1 deletion.

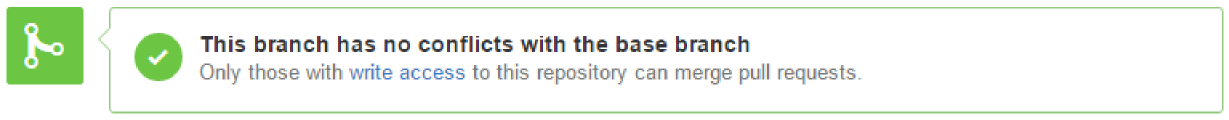
2 docs/moment/04-displaying/07-difference.md

~~~
@@ -26,7 +26,7 @@ a.diff(b, 'days') // 1

 The supported measurements are years, months, weeks, days, hours, minutes, and seconds. For ease of development, the singular forms
 are supported as of **2.0.0**. Units of measurement other than milliseconds are available in version **1.1.1**.

-By default, `moment#diff` will return number rounded down. If you want the floating point number, pass `true` as the third
 argument. Before **2.0.0**, `moment#diff` returned rounded number, not a rounded *down* number.
+By default, `moment#diff` will return a number rounded towards zero (down for positive, up for negative). If you want a floating
 point number, pass `true` as the third argument. Before **2.0.0**, `moment#diff` returned a number rounded to nearest, not a
 rounded number rounded towards zero.

 ```javascript
 var a = moment([2008, 6]);
~~~

Figure 61: Change details

By default, GitHub shows a Unified view of the change, but you can click the **Split** button to show the two files side by side, for another way to view the change.

One nice feature that GitHub reports to assist the repository owner is a status as to whether or not the pull request conflicts with the base branch.



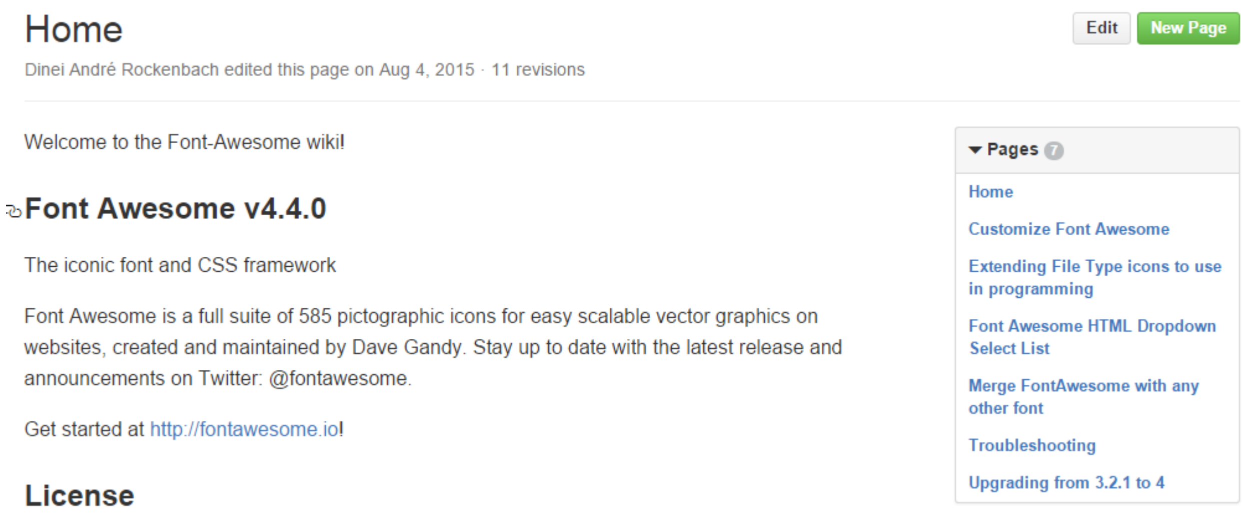
This branch has no conflicts with the base branch
Only those with write access to this repository can merge pull requests.

Figure 62: Conflict report

All of the information present as part of the pull request helps the repository owner decide whether or not to merge the request back to the main branch.

Wiki

GitHub allows you to create and maintain a Wiki about your repository, although this is an optional feature. If you have a Wiki, a tab will appear to let the repository viewer see the Wiki. A Wiki is a good source for ongoing documentation about a repository. The **readme.md** file provides a good introduction, while the Wiki can detail how to use the code in much more detail.



Home

Dinei André Rockenbach edited this page on Aug 4, 2015 · 11 revisions

Edit | New Page

Welcome to the Font-Awesome wiki!

Font Awesome v4.4.0

The iconic font and CSS framework

Font Awesome is a full suite of 585 pictographic icons for easy scalable vector graphics on websites, created and maintained by Dave Gandy. Stay up to date with the latest release and announcements on Twitter: @fontawesome.

Get started at http://fontawesome.io!

License

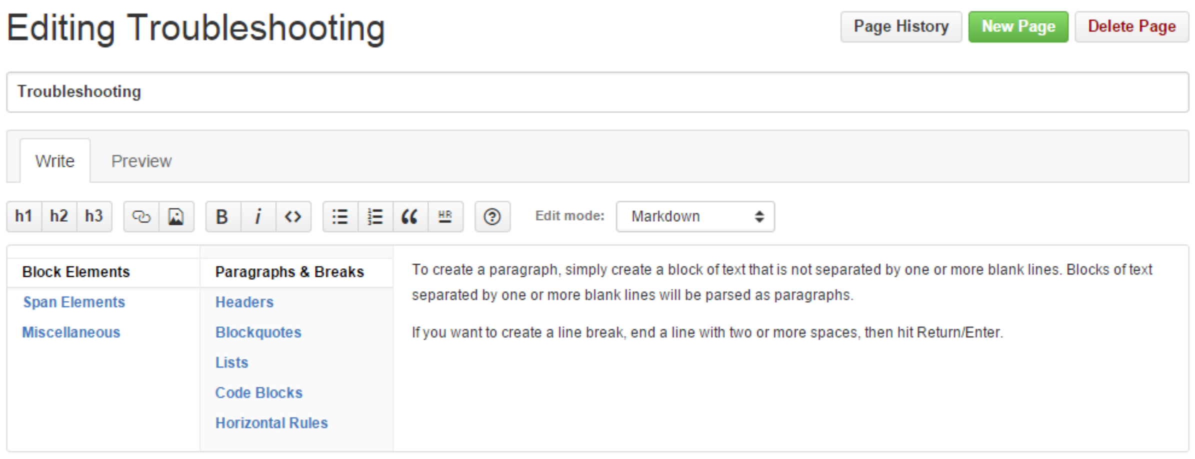
Pages 7

- Home
- Customize Font Awesome
- Extending File Type icons to use in programming
- Font Awesome HTML Dropdown Select List
- Merge FontAwesome with any other font
- Troubleshooting
- Upgrading from 3.2.1 to 4

Figure 63: Font Awesome Wiki

Wiki content is tracked, so you can review revisions to see what has changed in the Wiki over time. Wiki content and be updated just like any other repository, with pull requests to merge changes back into the Wiki.

Wiki editing provides a number of different formatting options, the default being Markdown, but many other Wiki editors are supported. The screen below shows the basic editor for Wikis.



Editing Troubleshooting

Page History | New Page | Delete Page

Troubleshooting

Write | Preview

Edit mode: Markdown

Block Elements	Paragraphs & Breaks
Span Elements	Headers
Miscellaneous	Blockquotes
	Lists
	Code Blocks
	Horizontal Rules

To create a paragraph, simply create a block of text that is not separated by one or more blank lines. Blocks of text separated by one or more blank lines will be parsed as paragraphs.

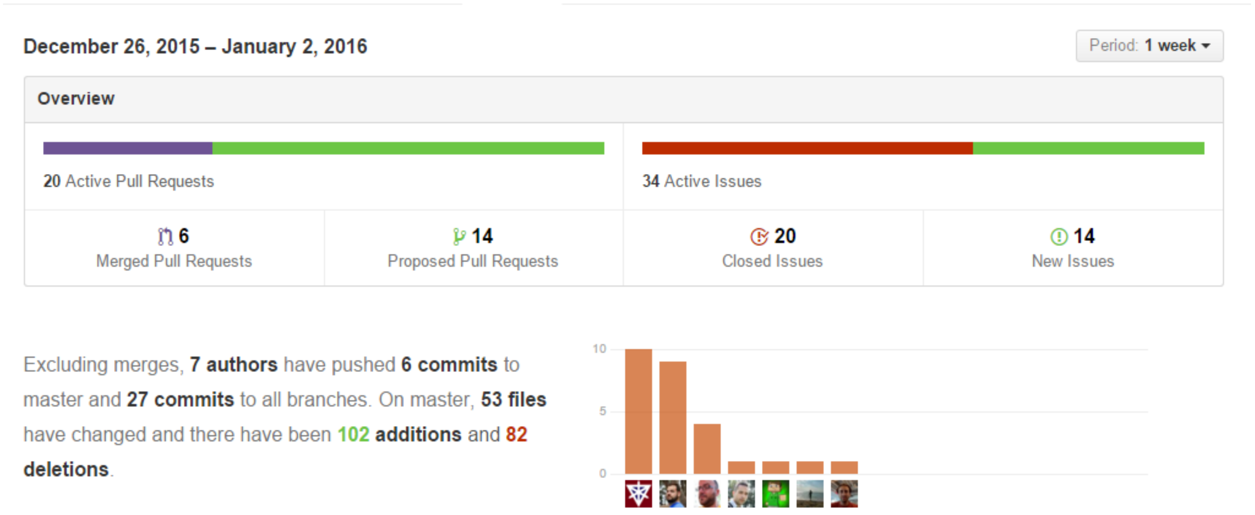
If you want to create a line break, end a line with two or more spaces, then hit Return/Enter.

Figure 64: Wiki editing

Wiki content is collaborative, allowing multiple users to enhance the Wiki and improve the documentation on the repository.

Pulse

The Pulse tab shows an overview of activity in the repository over a period of time (the default being the last week).


December 26, 2015 – January 2, 2016

Period: 1 week

Overview

20 Active Pull Requests

34 Active Issues

6 Merged Pull Requests	14 Proposed Pull Requests	20 Closed Issues	14 New Issues

Excluding merges, 7 authors have pushed 6 commits to master and 27 commits to all branches. On master, 53 files have changed and there have been 102 additions and 82 deletions.

Figure 65: Pulse

The top section shows the pull requests, both merged and proposed, and the active issues, both closed and new. You can click on any of the areas to get a list of all items making up the activity.

Below the summary is the information on the collaborators involved this activity period. You can click the picture below the graph bar to see that user's account page.

Finally, below the summary information at top is a list of all pull requests and issues within this activity period.

The Pulse tab gives you a sense of how active the repository is; this example represents one week's activity in Bootstrap.

Graphs

The Graphs tab shows a variety of data about the repository, its contributors, activity, etc. The menu across the top shows the various graphs that can be displayed:

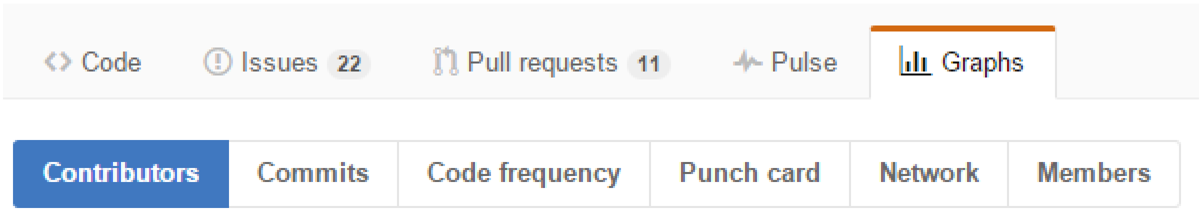


Figure 66: GitHub Graphs menu

The Contributors graph shows the most active contributors to the project and the commits they've made during the project's lifespan (it's the same page you get when you click on the contributors link while view the code).

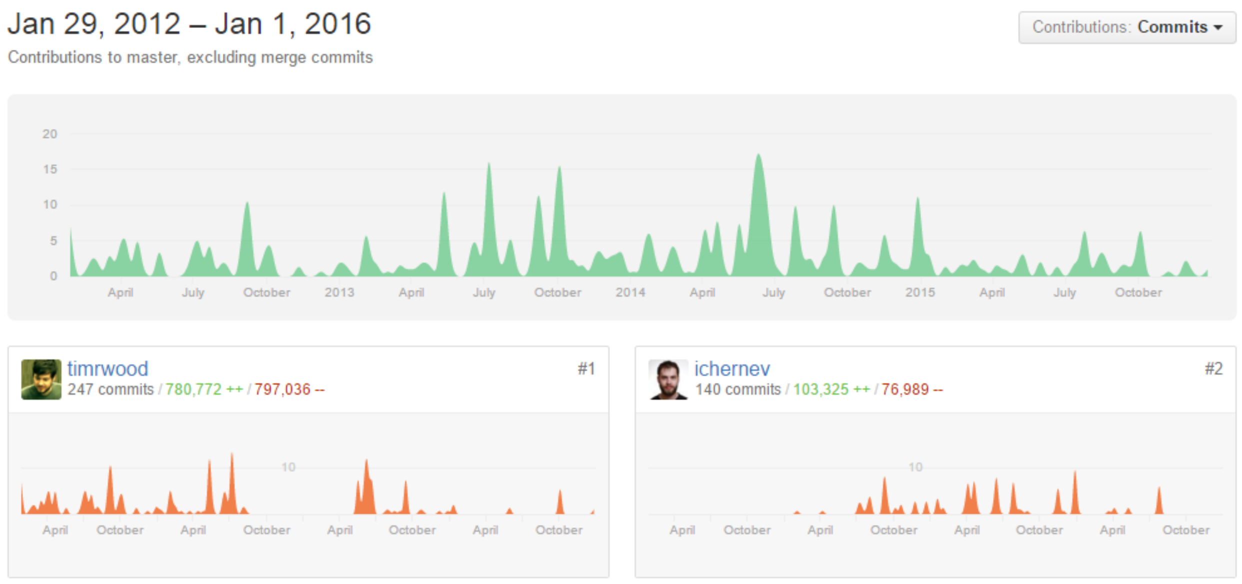

Jan 29, 2012 – Jan 1, 2016
Contributions to master, excluding merge commits
Contributions: Commits
timrwood
247 commits / 780,772 ++ / 797,036 --
#1
ichernev
140 commits / 103,325 ++ / 76,989 --
#2
April
July
October
2013
2014
2015

Figure 67: Contributors graph

The Commits graph shows two graphs; the top one is a bar graph of all activity to the project in the past year. The bottom graph shows the average number of commits per day.

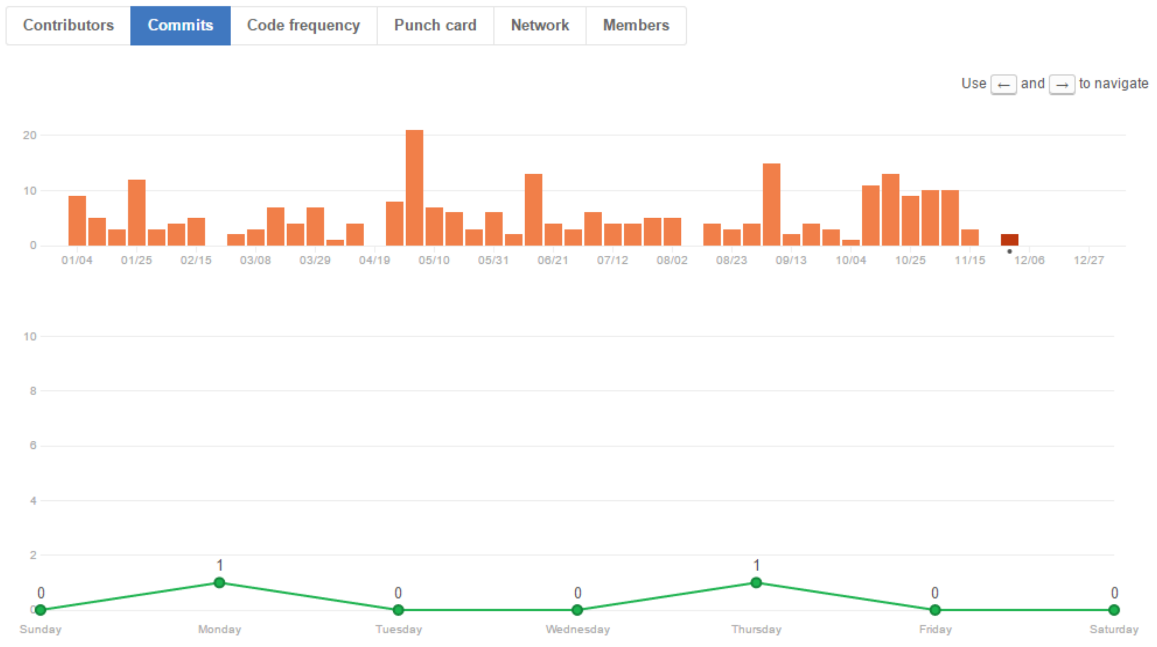
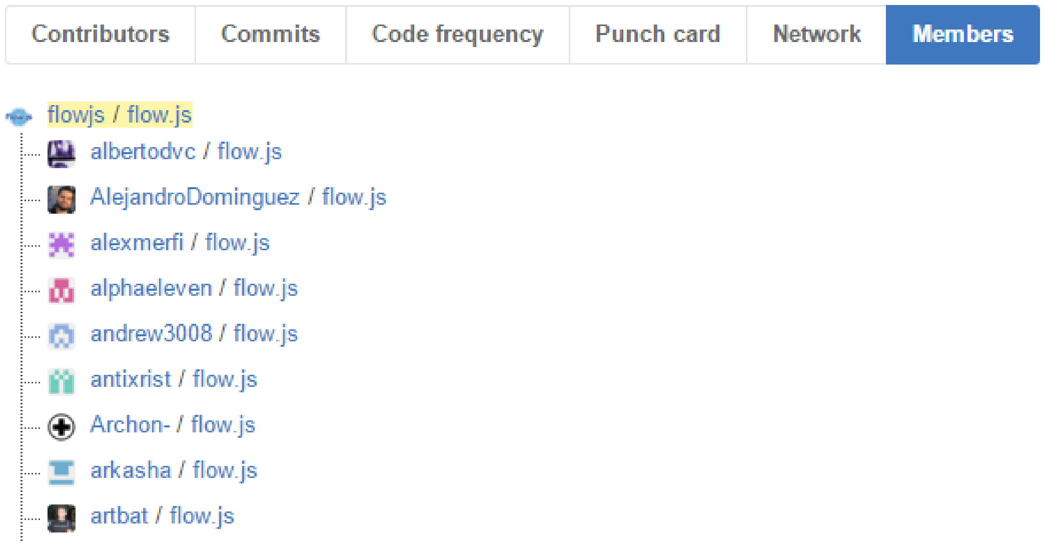


Figure 68: Commits graph

The top graph gives you a sense of how active the project is, while the bottom graph shows what days of the week get the most work done.

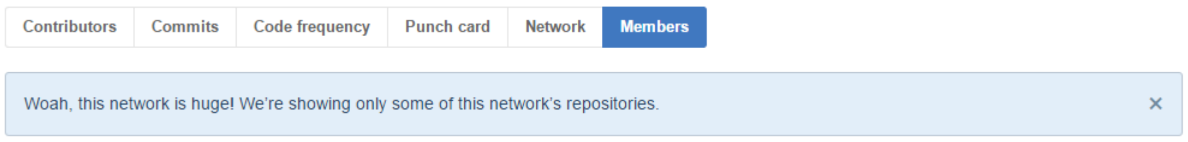
The Members graph shows all forks of this repository.; it shows the users and their forks. You can click on the user name to see that user's account details. You can also click on the fork name to visit that user's fork of the repository.



flowjs / flow.js
albertodvc / flow.js
AlejandroDominguez / flow.js
alexmerfi / flow.js
alphaeleven / flow.js
andrew3008 / flow.js
antixrist / flow.js
Archon- / flow.js
arkasha / flow.js
artbat / flow.js

Figure 69: Members graph

Sometimes, if the fork/contributor list is too large, GitHub will show its colloquial sense of humor, and only show some of the forks.



Woah, this network is huge! We're showing only some of this network's repositories.

Figure 70: Bit too big

There are other graphs available to show statistics about the repository and its activity. You can visit this link to read about all of the graphs.

readme.md contents

Below the list of files in the repository is the content of the **readme.md** file. This file should provide a good overview of what the repository is about, how to install and use it, etc. You will find all kinds of readme files on GitHub; some are complete user manuals, while others are a couple lines telling you how to install the software.

At a minimum, I would provide a short (one-paragraph) summary of the code, and how to get it installed (minimum requirements). This at least can whet the appetite of the user. You can then either provide much more detail (in the remainder of the readme file) or as a separate documentation file in your repository.

Summary

There’s a lot of detail in this chapter about the information GitHub provides about the repositories. You should be comfortable knowing how to use this information to help decide if a particularl repository can help your application, and if you can contribute back to the code base.

In the next chapter, we will talk about how to obtain the code for either your own personal use or with the intent of getting involved with the project.

Chapter 10 Obtaining the Code

You've searched GitHub, reviewed all the results, and found a repository that should help. The next step is to download the code. If you are only planning to use the code, without collaborating and contributing to make the code better, you can download the code to your local machine.

Download ZIP

The easiest way to grab the entire repository is to click the **Download ZIP** button on the menu bar. Using this method, you'll download only the source and other files (no repository history or details).

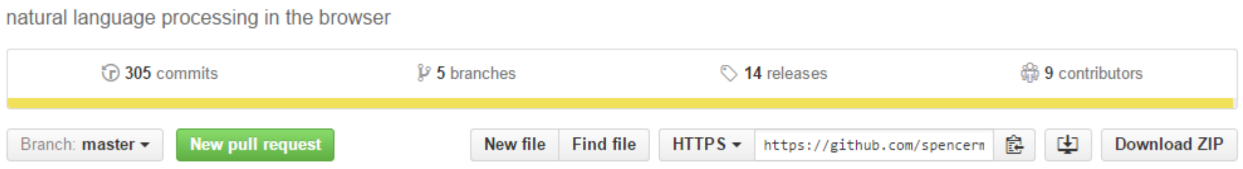
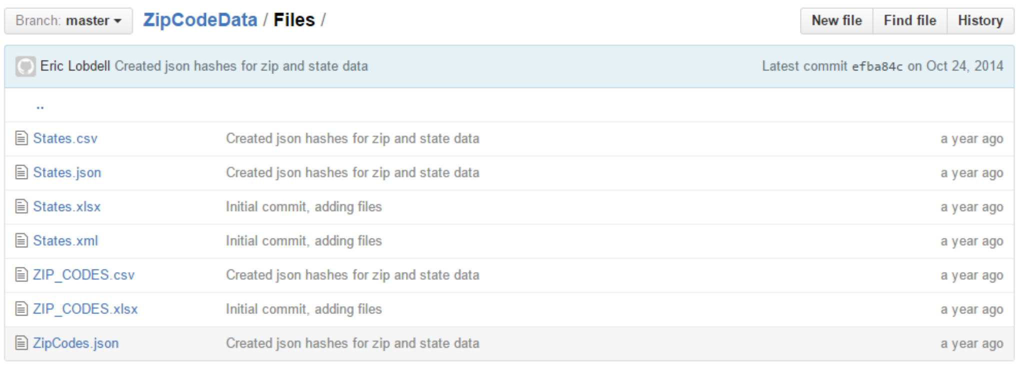


Figure 71: Download ZIP file

One item to note is that the ZIP will contain all of the files, including the **readme.md** file. This file may contain imbedded markdown codes, so it might not be easily readable in a text editor.

Downloading individual files

You can also download individual files by clicking on the file name. GitHub is very smart about determining how to display the file contents, particularly of text or programming language files. Binary files are generally not shown, but you can click on them to download them. As example, let's look at ZipCodeData. Say you need zip code and state information, and your search leads you to this repository. When you click on the Files subfolder, you see the following results.

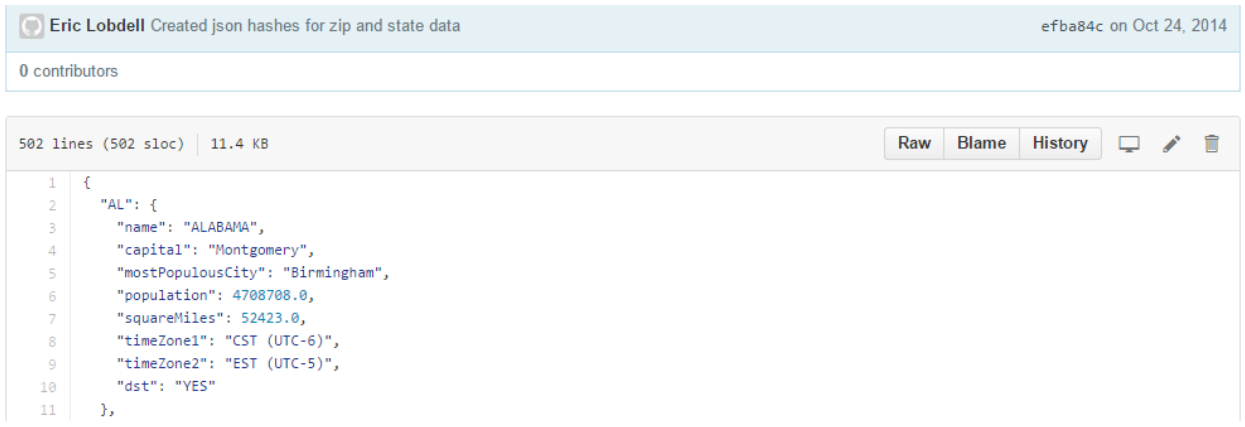


Eric Lobdell Created json hashes for zip and state data — Latest commit efba84c on Oct 24, 2014

..		
States.csv	Created json hashes for zip and state data	a year ago
States.json	Created json hashes for zip and state data	a year ago
States.xlsx	Initial commit, adding files	a year ago
States.xml	Initial commit, adding files	a year ago
ZIP_CODES.csv	Created json hashes for zip and state data	a year ago
ZIP_CODES.xlsx	Initial commit, adding files	a year ago
ZipCodes.json	Created json hashes for zip and state data	a year ago

Figure 72: Zip code files folder

However, you don't need the entire solution and other data files; you are only interested in the JSON-formatted data. When you click on the file name, the file is shown in a GitHub window.


Eric Lobdell Created json hashes for zip and state data
efba84c on Oct 24, 2014
0 contributors
502 lines (502 sloc) | 11.4 KB
Raw
Blame
History

```
{
  "AL": {
    "name": "ALABAMA",
    "capital": "Montgomery",
    "mostPopulousCity": "Birmingham",
    "population": 4708708.0,
    "squareMiles": 52423.0,
    "timeZone1": "CST (UTC-6)",
    "timeZone2": "EST (UTC-5)",
    "dst": "YES"
  },
```

Figure 73: Zip code JSON data

When you click on the **Raw** tab, GitHub opens the file with no adornments around it. You can then use the browser context menu and save the contents to your local drive.

The Blame tab (despite its name), is a handy way to identify who made changes in a file and provides a brief description of why the change was made.

Forking the project

Although you can easily download files and code, if a project piques your interest, you should consider creating a fork, and possibly contributing to the project by reviewing some of the open issues. The essence of open source software is collaboration, and most repository owners will welcome assistance on their projects.

If you decide you want to work on a project, you can click on the **Fork** icon at the top of repository.

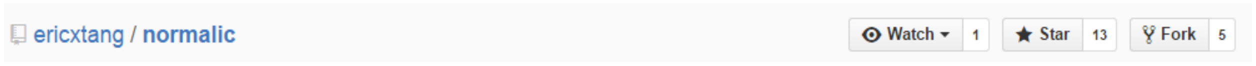
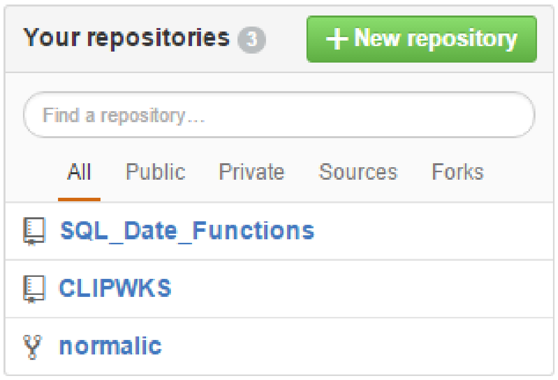


Figure 74: Fork a project

Once you fork the project, your repository list will now show the Forked repository, indicated by the forked icon.



SQL_Date_Functions

CLIPWKS

normalic

Figure 75: Showing the forked repository

You now have a copy of the repository, and can start reviewing the issues to see if you might want to contribute. In Figure 75, normalic is a handy library for parsing U.S. address data, but written in Ruby. My goal in the fork is to create a C# version for my own use, and then put it back onto GitHub (crediting the original repository as well).

Summary

You might use GitHub simply as a large library of source code and data, and no doubt will find many useful repositories available. However, in the spirit of open source software, you might find yourself helping improve other works in collaboration to produce some better software.

Chapter 11 Collaboration

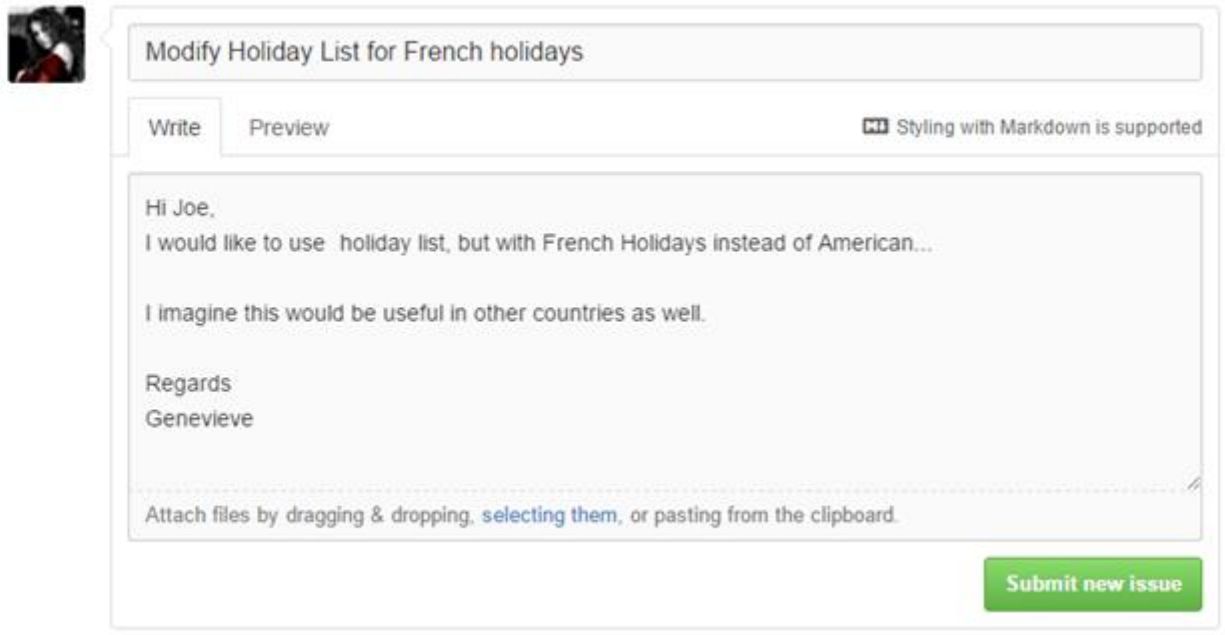
In the earlier chapters, we focused on searching GitHub and downloading project files. In the next couple of chapters, we're going to focus on collaboration, how to create forks, and how to deal with pull requests.

SQL Date functions

Early on in this book, we created a repository of some SQL code dealing with dates. One of the scripts (**`holiday list`**) would create a list of holidays for a given year. Genevieve, a programmer in France, likes the script, but she wants to revamp it for French holidays, not just American ones. Rather than simply downloading the script and modifying it for her own use, she decides to submit an issue, suggesting holidays for different countries.

Submitting an issue

Genevieve opens the repository in GitHub and clicks the **Issues** tab. She clicks **New Issue** to submit a new issue about the code.


Write
Preview
Styling with Markdown is supported
Attach files by dragging & dropping, selecting them, or pasting from the clipboard.
Submit new issue

Modify Holiday List for French holidays

Hi Joe,
I would like to use holiday list, but with French Holidays instead of American...

I imagine this would be useful in other countries as well.

Regards
Genevieve

Figure 76: Submit issues

Note that she can use Markdown syntax to control her issue appearance, and can upload files related to the issue as well.

Joe receives the email

Since I am watching my own repository, I received the email from GitHub letting me know that an issue has been submitted.

Hi Joe,
I would like to use holiday list, but with French holidays instead of American holidays.

I imagine this would be useful in other countries as well.

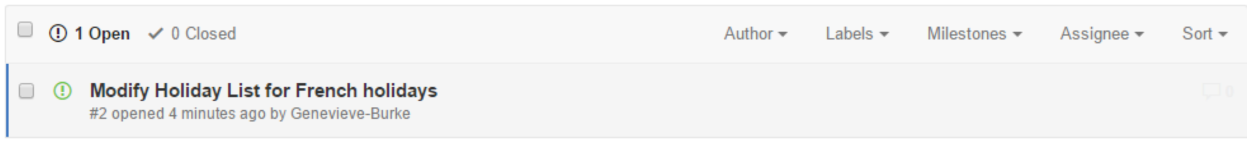
Regards,
Genevieve

—
Reply to this email directly or view it on GitHub.

I think it's an excellent suggestion, but I'm not sure I have the time to research holidays for multiple countries.

Viewing the issue

I open up GitHub and log in, and I can view the issues associated with this repository. So far, this is the only issue reported.



Modify Holiday List for French holidays
#2 opened 4 minutes ago by Genevieve-Burke

Figure 77: Issues

Since this is the only issue so far, and it will take some research or assistance to get the holidays from other countries, I decide to create a new branch. I'll call this branch RELEASE and use it to do the work for adding country support. If any bugs get reported in the existing base, I can correct those without giving users any partially completed code for language support.

Creating the branch

A branch represents a totally separate copy of all of the code in the repository. This allows you or any collaborators to work on this code without impacting the main branch. Once you've completed your code, GitHub will help you merge it back into the main branch if you choose to.

To create the branch, click on **Branch** in the repository and type in the name of the branch.

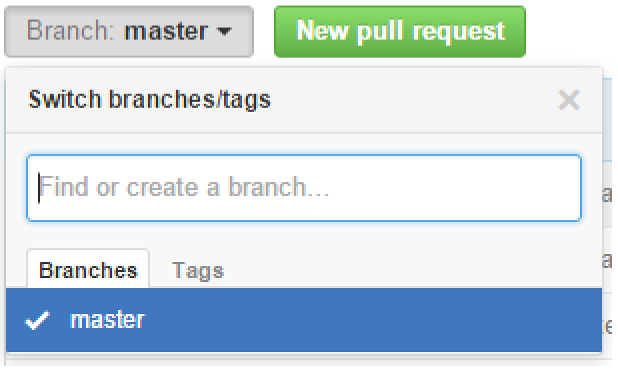


Figure 78: Creating a new branch

GitHub will see that the branch does not exist, and offer to create a new branch for you from the currently selected branch (master in this example).

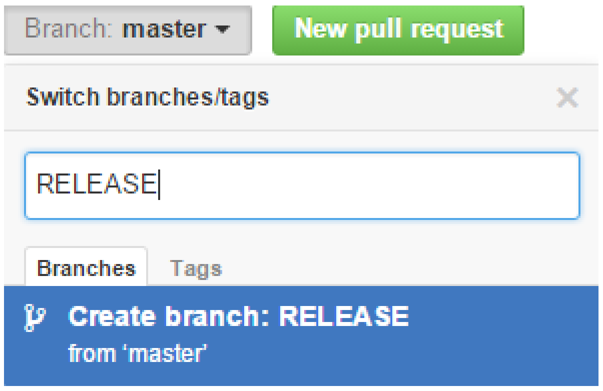


Figure 79: New branch

You will now have a separate copy of the code so that you can work on it without disturbing the main code branch.

Adding a collaborator

Since Genevieve has agreed to help out, I update the repository settings and add her as a collaborator.

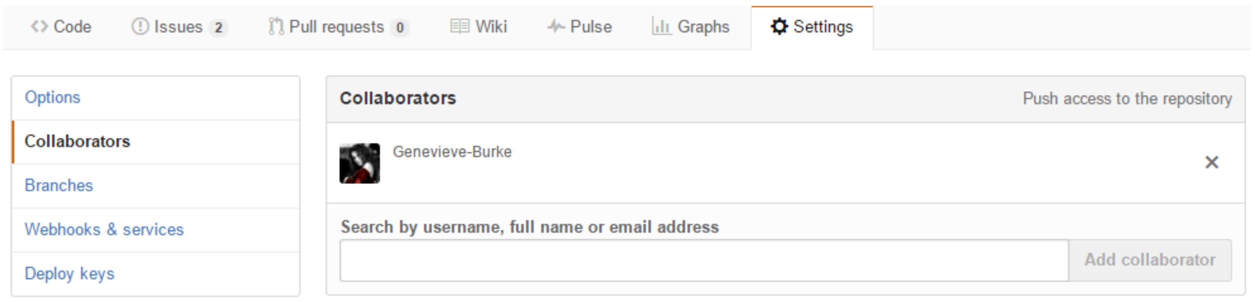


Figure 80: Adding a collaborator

At this point, Genevieve can create a fork from the Dev branch of the repository. She creates the fork and now has her own copy to work from.

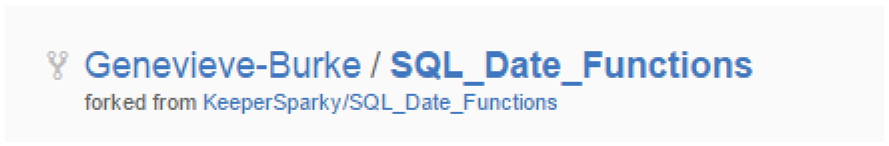
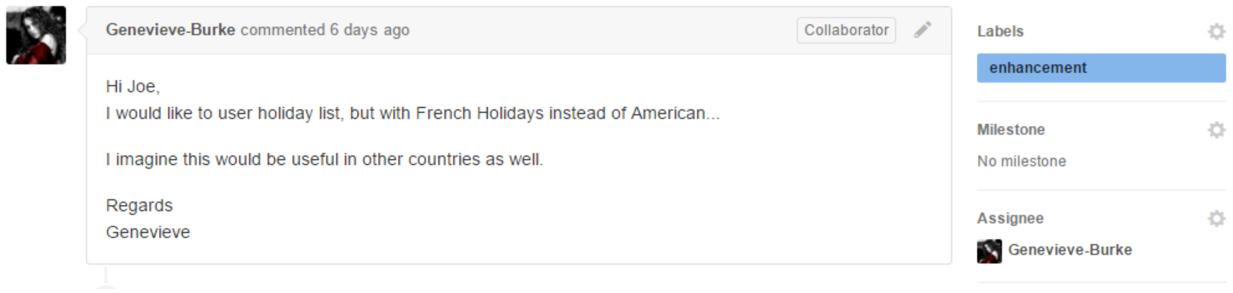

Genevieve-Burke / SQL_Date_Functions
forked from KeeperSparky/SQL_Date_Functions

Figure 81: Forked copy of repository

At this point, Genevieve can start making her changes to implement French holidays in the SQL user-defined function.

Assigning Genevieve to the issue

Joe now goes to the Issue screen and assigns the issue to Genevieve.



Hi Joe,
I would like to user holiday list, but with French Holidays instead of American...

I imagine this would be useful in other countries as well.

Regards
Genevieve

Figure 82: Assigning the issue

Summary

At this point, an issue has been submitted and the user agrees to collaborate on fixing the issue. The repository owner (Joe) has taken three steps: created a branch, made Genevieve a collaborator, and assigned the issue to her.

The collaborator (Genevieve) has created her own fork so she can make code updates.

And time passes…

Chapter 12 Pull Request

At some point during the collaboration, a contributor is likely to have completed and tested the code well enough that she feels the code could be added to the main branch. Genevieve decides to create the pull request by click on the **Pull request** button.

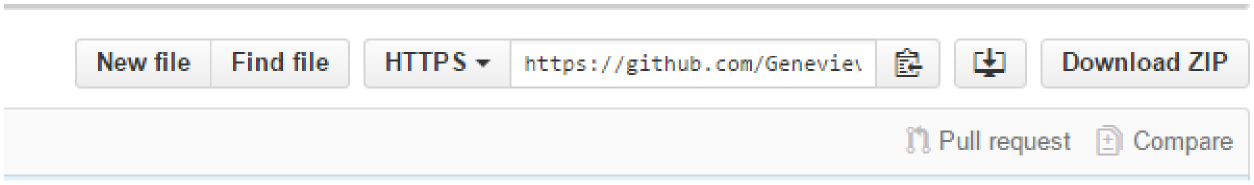
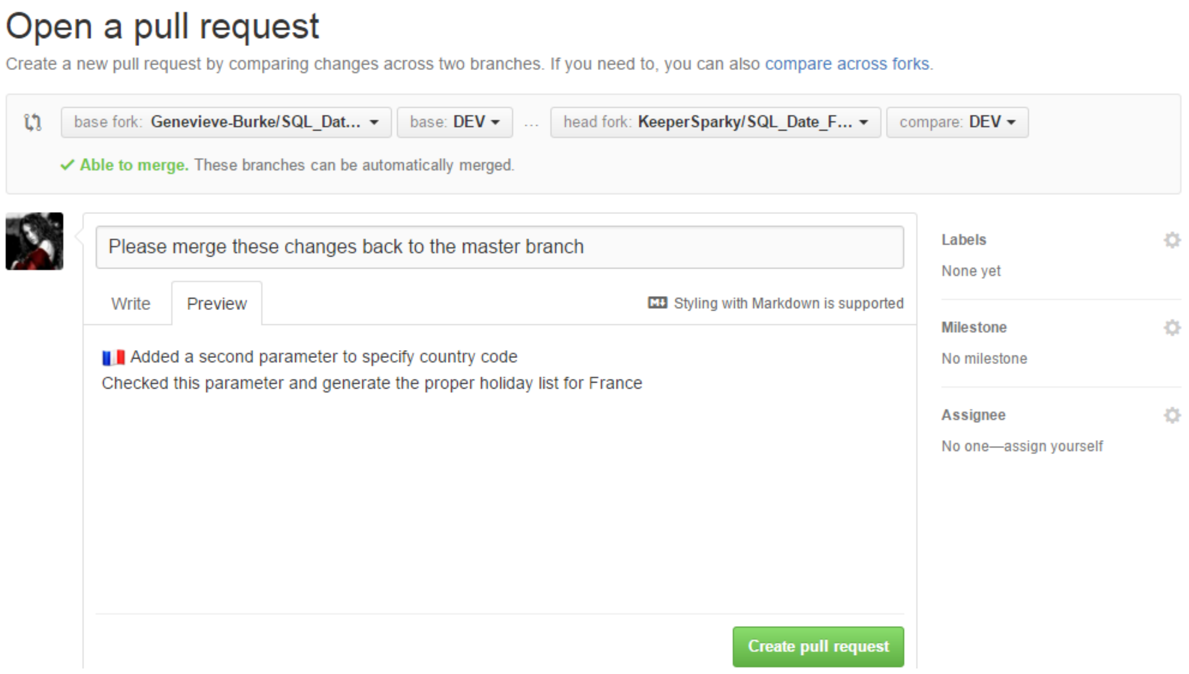


Figure 83: Pull request

Creating the request

Genevieve submits the request and adds a description of the work she has done.



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: Genevieve-Burke/SQL_Dat... base: DEV ... head fork: KeeperSparky/SQL_Date_F... compare: DEV

✓ Able to merge. These branches can be automatically merged.

Please merge these changes back to the master branch

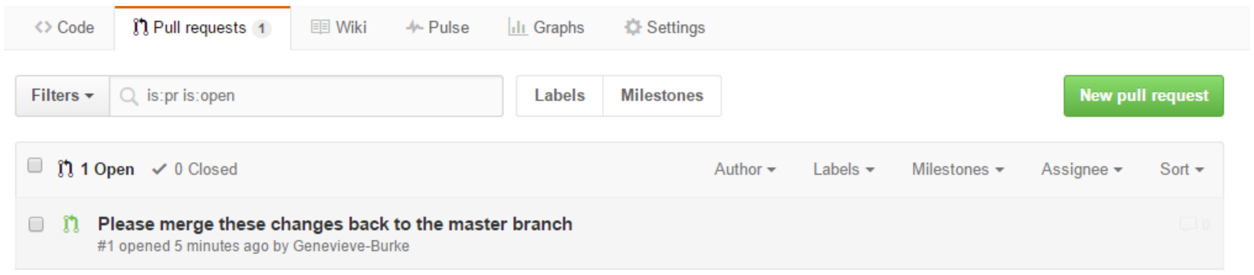
Added a second parameter to specify country code
Checked this parameter and generate the proper holiday list for France

Figure 84: Pull request details

The repository owner will now see the pull request when he opens the repository. The owner would have also received an email that a pull request has been made.

Reviewing the request

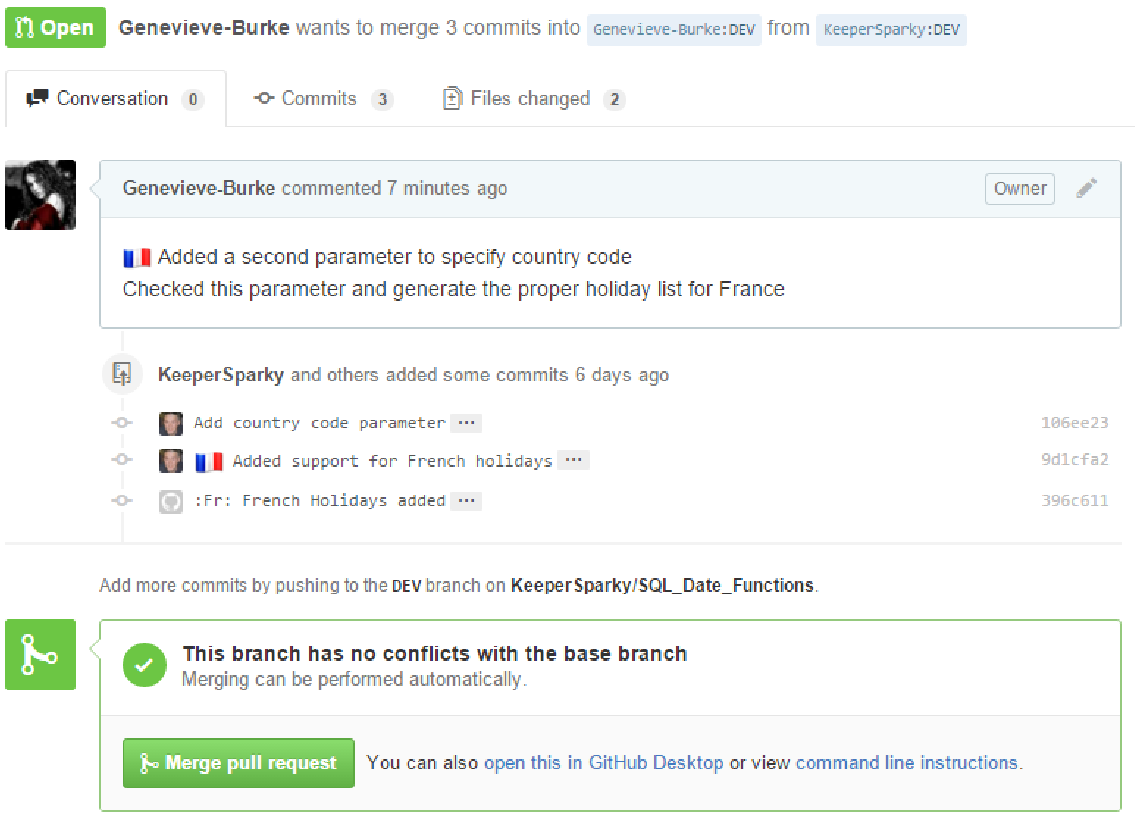
The owner sees the request when he opens the repository.



Please merge these changes back to the master branch
#1 opened 5 minutes ago by Genevieve-Burke

Figure 85: Pull request view

By clicking on the pull request itself, the details of the request are displayed.



Genevieve-Burke wants to merge 3 commits into `Genevieve-Burke:DEV` from `KeeperSparky:DEV`

Genevieve-Burke commented 7 minutes ago

🇫🇷 Added a second parameter to specify country code
Checked this parameter and generate the proper holiday list for France

KeeperSparky and others added some commits 6 days ago

```
Add country code parameter ...                 106ee23
🇫🇷 Added support for French holidays ...        9d1cfa2
:Fr: French Holidays added ...                 396c611
```

Add more commits by pushing to the DEV branch on KeeperSparky/SQL_Date_Functions.

This branch has no conflicts with the base branch
Merging can be performed automatically.

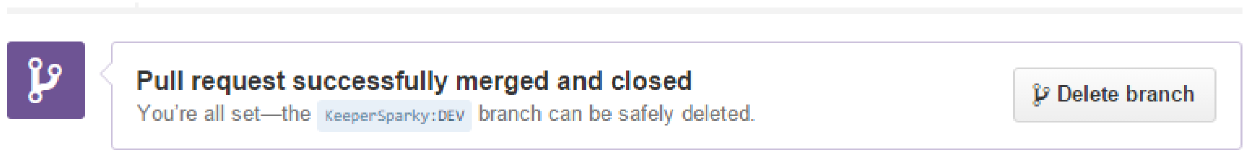
You can also open this in GitHub Desktop or view command line instructions.

Figure 86: Pull request details

The owner can now look at the commits and actual file changes to decide whether or not to merge the code. After reviewing the changes and accepting them, the owner clicks on the Merge pull request button. GitHub will ask for a confirmation, and once the confirmation is made, will perform the merge.

Successful merge

When the merge is successful, GitHub will report it and update the repository:



Pull request successfully merged and closed

You're all set—the `KeeperSparky:DEV` branch can be safely deleted.

Figure 87: Successful pull request

Since the pull request is complete, it will now appear as a closed pull request when the repository is being viewed.

Summary

Once the collaborator completes the requests, and they request that the code be merged back in, the owner can review the changes and accept them (perform the merger), or perhaps ask the collaborator questions, or identify additional work that needs to be done before accepting the changes. GitHub provides all the tools and comparisons that the owner needs to make a decision to include the new code.

Final Thoughts

Git is a powerful distributed version control tool, and GitHub a great way to store your repository on the web. With its version control ability, it can be a handy tool for any programmers or developers working with files that have changes to be tracked. However, the real beauty of GitHub is its issue tracking and collaboration tools to support the spirt of open source software.

You can use GitHub simply as a massive library of code to assist you in your development efforts, or you can be an active member of the open source community by participating and collaborating in projects that strike your interest.

Enjoy the world of GitHub!

Appendix 1 GitHub Markdown

Most text entered into GitHub can take advantage of the Markdown syntax for formatting text. Markdown text is easier to read and write than the equivalent HTML, but can be displayed as nicely formatted HTML within the GitHub site.

Basic Markdown commands

Any text entered into a Markdown window is consider a paragraph. Each blank line begins a new paragraph. You can write the entire content simply by typing, and the Markdown engine within GitHub will determine where to add the **`<p>`** paragraph tags when displaying the HTML results.

Headings

You can use the **`# character`** to create a heading in the text. All text between the **`#`**'s will become an HTML **`<h1>`** tag. You can create **`<H2>`** through **`<H6>`** tags by increasing the number of **`#`**'s around text.

```
# This is H1 text#

### This is H3 text ###
```

Bold/Italic text

Any text surrounded by an asterisk character (**`*`**) will appear in *italics.* Any text surrounded by two sequential asterisks (**`**`**) will appear in **bold.** You can use the underscore character (_) if you need to nest bold and italic. For example:

```
**This should be bold with _italic_ inside**
```

This will appear as the following when view in GitHub:

This should be bold with *italic* inside.

Quotes

If you want add a quote, you can place a **`>`** symbol before the quote. For example:

Abraham Lincoln once said

>Don't believe everything you read on the Internet

This will appear as follows when displayed in GitHub:

> Abraham Lincoln once said
>
> | Don't believe everything you read on the Internet

Good advice, by the way!

Lists

Lists can be made by placing a dash (**-**) or asterisk (*****) prior to the item. For example:

```
* master branch

* January sprint

* release branch
```

Will appear as follows in GitHub:

- master branch
- January sprint
- release branch

You can use numbers rather than the dash or asterisk to create an ordered list. Lists can also be nested, and ordered and unordered lists can be combined.

Code formatting

The backtick character (**`**) will display the text between backticks as code, such as:

Press the `ENTER` key

If you put triple backticks around a section of text, that entire section will appear as code. For example:

~~~
```

if (x==1) {

      Console.write("EXPECTED MULITPLE ROWS");

}

```
~~~

This will look like a code block when displayed in GitHub:

```
if (x==1) {
        Console.write("EXPECTED MULTIPLE ROWS")
}
```

This allows code to stand out from the rest of the text.

Links

Links (URLs) and images (URLs referencing an image file) can be imbedded by enclosing the link text in brackets **`[ ]`** and the referencing URL in parentheses **`( )`**. To reference GitHub, for example, you might use the following text.

`[Visit my GitHub page] (https://www.github.com/KeeperSparky)`

This will produce the following link in the GitHub display:

Visit my GitHub page

This link will navigate the user to the indicated page when clicked.

Markdown

Markdown has been around for over 10 years, and you can visit this link if you want to read more about.

GitHub Markdown

In addition to the standard Markdown constructs, GitHub has its own additional enhancements for the GitHub environment.

Underscores

The underscore character in GitHub is not used to combine **bold** and *italic* text, since underscores are often used in code. GitHub will ignore underscore characters between words and treat them as part of the text.

URLs

If GitHub detects a URL pattern, such as **`http://github.com`** , it will automatically treat it as a hyperlink. You can still name the hyperlink using the Markdown syntax, but if you want to display a URL, it is much easier just to type the URL in directly and let GitHub display it.

Strikethrough text

Since additions and deletions are a big part of GitHub code displays, you can use the tilde characters (**~~**) to indicate strikethrough text. For example:

```
~~Removed currency symbol~~
```

This syntax displays as the following in GitHub:

~~Removed currency symbol~~

Syntax highlighting

You can use the triple backtick syntax to display a code section, but GitHub adds an extra twist. By specifying a language name after the first backtick header, GitHub will color-code keywords and provide other syntax-highlighting features. By adding JavaScript as the language, GitHub renders this code as:

~~~
```JavaScript

if (x==1) {

      Console.write(“EXPECTED MULITPLE ROWS”);

}
~~~

In GitHub:

```
if (x==1) {
        Console.write("EXPECTED MULTIPLE ROWS")
}
```

This helps the developers, since the code probably looks similar to their development tools.

Users and issues

GitHub also allows the **#** to be used to link to an existing issue number, and the **@** to specify a user name.

Summary

The combination of Markdown and GitHub’s enhancements provides a rich environment for making nicely styled text in issues, commit messages, Wiki text, etc. Once you get comfortable with the basic syntax, you’ll find it second nature to write that text.

Appendix 2 Emoji

In addition to using Markdown, you can add Emoji characters into your GitHub text boxes. Emoji are mini pictures that get displayed instead of text in the body of your document. Emoji names are delimited by the colon character. If you type a colon within a text window, GitHub will bring up the available Emoji beginning with the letter after the colon:

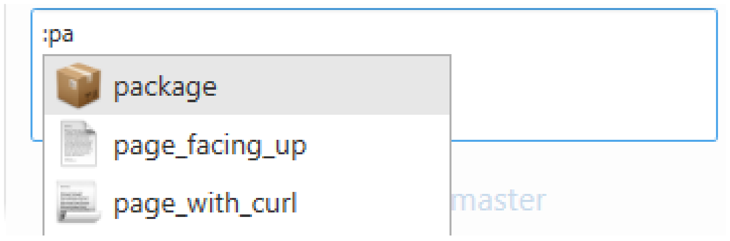
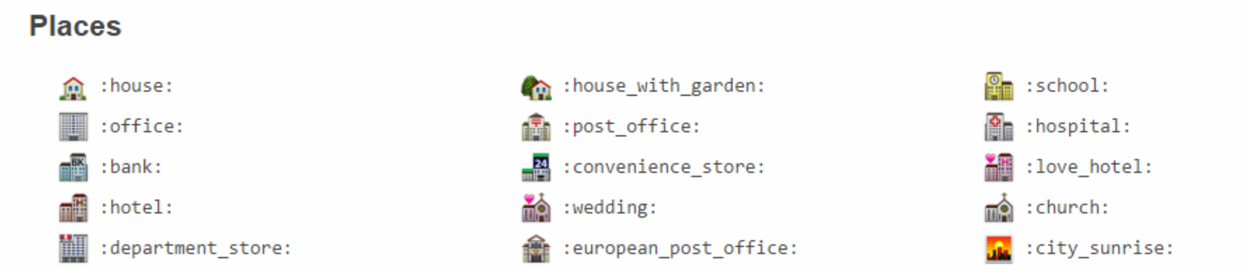


Figure 88: Emoji selection

Emoji can make your text easier to read and understand, and just more fun to look at. Some sample Emoji are shown in Figure 89:



Places

:house:	:house_with_garden:	:school:
:office:	:post_office:	:hospital:
:bank:	:convenience_store:	:love_hotel:
:hotel:	:wedding:	:church:
:department_store:	:european_post_office:	:city_sunrise:

Figure 89: Sample Emoji

Emoji is used by many websites, not just GitHub. To view the complete set, visit this website.