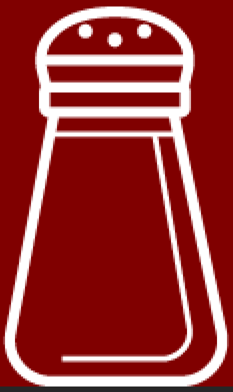


GitHub

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.

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.

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.

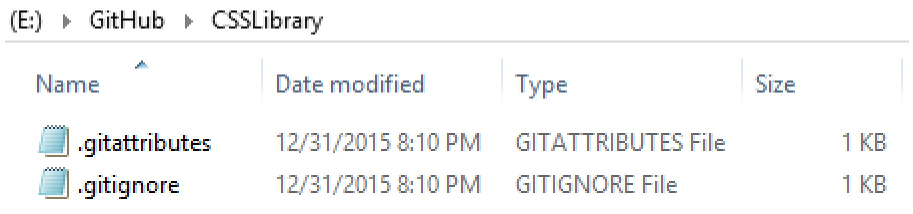
Working directory

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.

APPROVED

Appendix 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:

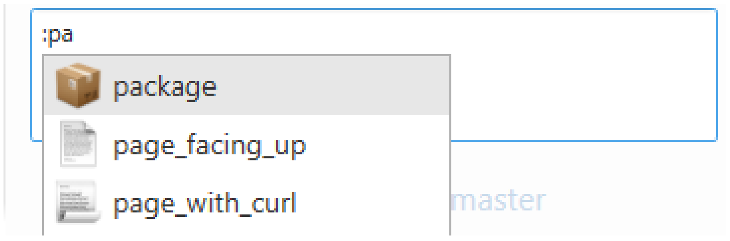
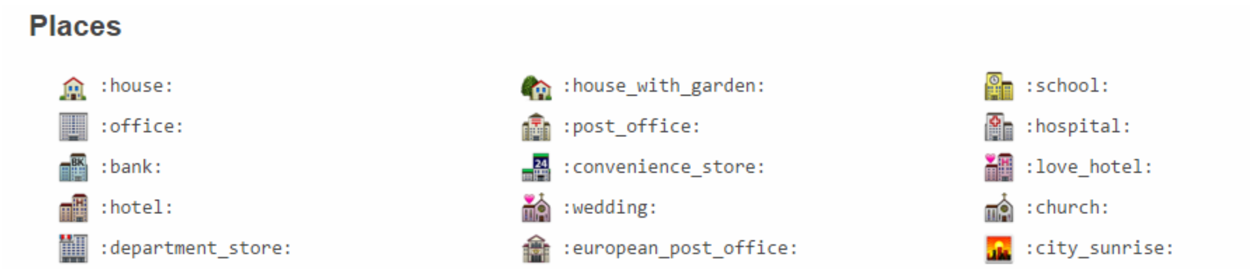


Figure 88: Emoji selection

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.