GIT + GITHUB

Abstract

A condensed manual on what Git and GitHub are and how to use these powerful tools.

Any Git commands shown, will be italicized in blue

These Git commands must be entered as shown, if not, commands will not be executed.

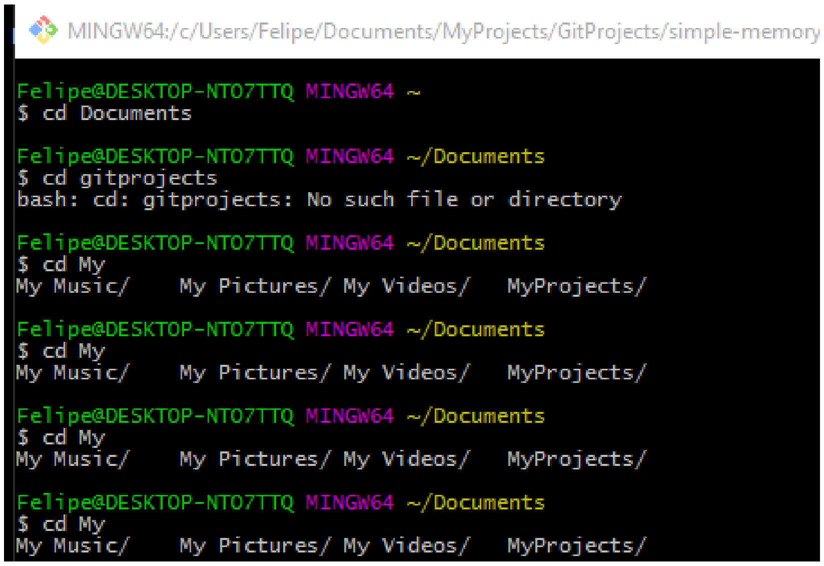
Due to this book being a condensed version, it will not cover absolutely everything. It will mainly cover the basics.

This book is written in the style of an instruction manual. You may skip around section(s) if you have these procedures completed prior to reading this book.

Table Of Contents

1. What is Git?

Git is a powerful Version Control based system that has a User Interface similar to Command Prompt (Windows) or Terminal (Mac). Used in order to create version controlled workspaces called *Repositories*.



```
MINGW64:/c/Users/Felipe/Documents/MyProjects/GitProjects/simple-memory

Felipe@DESKTOP-NTO7TTQ MINGW64 ~
$ cd Documents

Felipe@DESKTOP-NTO7TTQ MINGW64 ~/Documents
$ cd gitprojects
bash: cd: gitprojects: No such file or directory

Felipe@DESKTOP-NTO7TTQ MINGW64 ~/Documents
$ cd My
My Music/    My Pictures/ My Videos/   MyProjects/

Felipe@DESKTOP-NTO7TTQ MINGW64 ~/Documents
$ cd My
My Music/    My Pictures/ My Videos/   MyProjects/

Felipe@DESKTOP-NTO7TTQ MINGW64 ~/Documents
$ cd My
My Music/    My Pictures/ My Videos/   MyProjects/

Felipe@DESKTOP-NTO7TTQ MINGW64 ~/Documents
$ cd My
My Music/    My Pictures/ My Videos/   MyProjects/
```

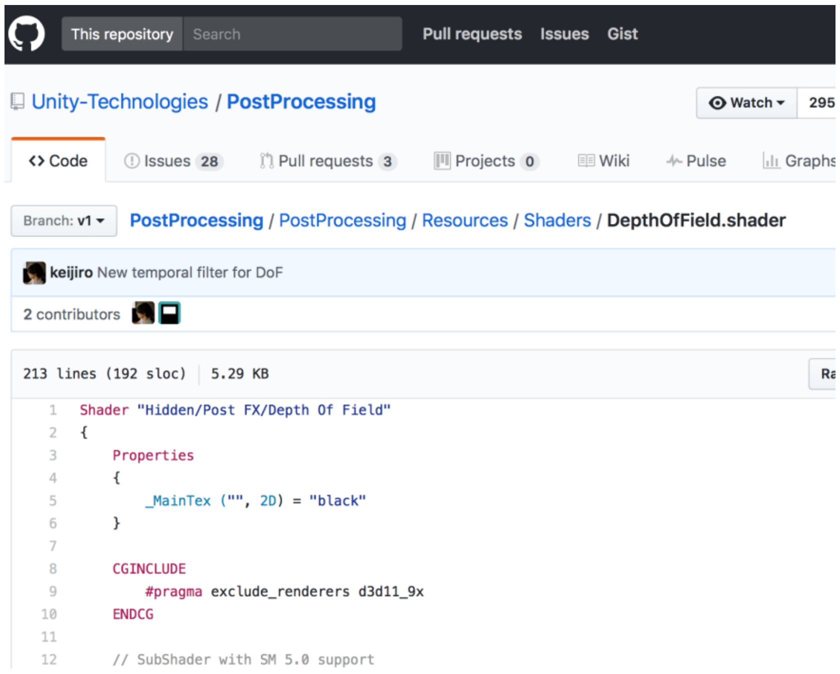
Git uses specialized key words in order to execute a task. May or may not be the same key words used inside of Command Prompt (Windows) or Terminal (Mac). These key words give you the ability to create, modify, delete your directories however you see fit.

- Basic Git commands will be included on a later page in this manual

2. What is GitHub?

GitHub is an online platform that enables developers to put their source code somewhere for storage and code management purposes.

- **Public Repositories** are seen as open source projects. Anyone can see and use your source code. Only authorized developers can contribute. Public Repository are free.


This repository
Search
Pull requests
Issues
Gist
Unity-Technologies / PostProcessing
Watch
295
Code
Issues 28
Pull requests 3
Projects 0
Wiki
Pulse
Graphs
Branch: v1
PostProcessing / PostProcessing / Resources / Shaders / DepthOfField.shader
2 contributors
213 lines (192 sloc) | 5.29 KB

keijiro New temporal filter for DoF

```
Shader "Hidden/Post FX/Depth Of Field"
{
    Properties
    {
        _MainTex ("", 2D) = "black"
    }

    CGINCLUDE
        #pragma exclude_renderers d3d11_9x
    ENDCG

    // SubShader with SM 5.0 support
```

- **Private Repositories** are projects that are closed off only to those working on it. (example: source code to Microsoft Word is only seen by Microsoft Developers/Software Engineers). Private Repositories come at a cost.

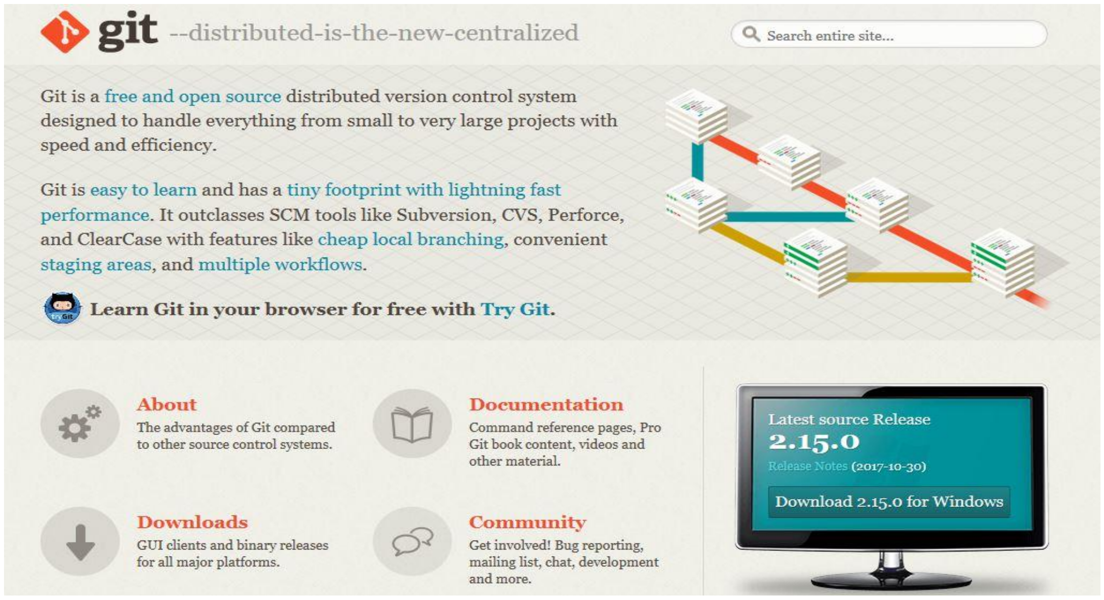
Uses:

- Identifying where software bugs occur
- Identify who is responsible for what source code changes
- Identify what code was added, modified, and deleted between versions
- Keep archive of all of versions made to project (commits)

3. Getting Started (Git)

3.1 Download & Install Git

First things first… Download Git.



Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Learn Git in your browser for free with Try Git.

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

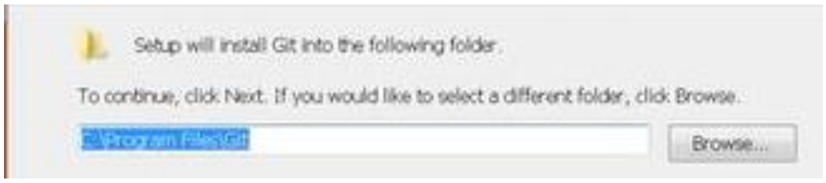
Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.15.0
Release Notes (2017-10-30)
Download 2.15.0 for Windows

In order to download Git, visit https://git-scm.com/. Once there, you will need to download the correct version of the program. Meaning that you need to download the Windows or Mac files (whichever system you plan to run Git on).

3.2 Installation Process

1. Open Installation Wizard
2. Select Installation Location for Git.

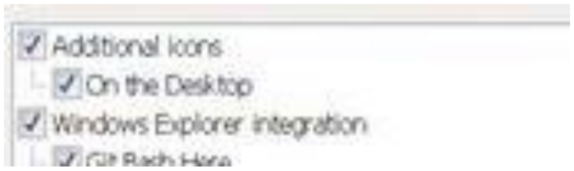

Setup will install Git into the following folder.

To continue, click Next. If you would like to select a different folder, click Browse.

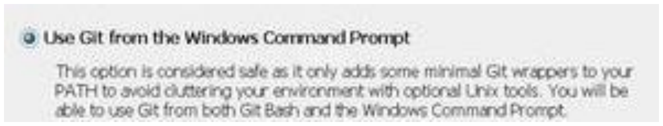
C:\Program Files\Git

Browse...

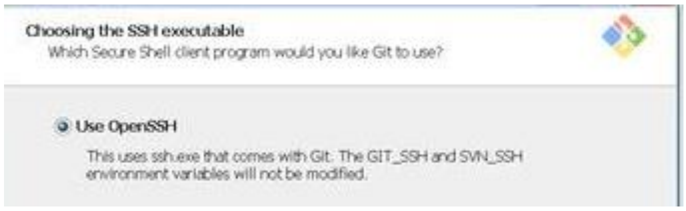
3. **Enable Additional Icons** on the Desktop.


✓ Additional icons
✓ On the Desktop
✓ Windows Explorer integration
✓ Git Bash Here

4. When choosing your PATH environment, select the 2nd option: ***Use Git from the Windows Command Prompt***


● Use Git from the Windows Command Prompt

This option is considered safe as it only adds some minimal Git wrappers to your PATH to avoid cluttering your environment with optional Unix tools. You will be able to use Git from both Git Bash and the Windows Command Prompt.

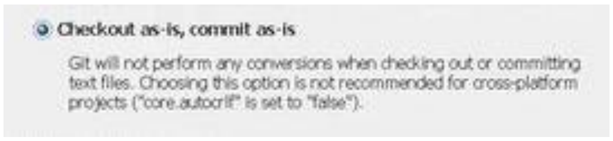

Choosing the SSH executable

Which Secure Shell client program would you like Git to use?

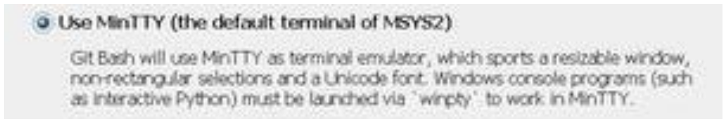
● Use OpenSSH

This uses ssh.exe that comes with Git. The GIT_SSH and SVN_SSH environment variables will not be modified.

5. When choosing the SSH executable, select ***Use OpenSSH***
6. When choosing to configure line ending conversions, select the last option: ***Checkout as-is, commit as-is***


● Checkout as-is, commit as-is

Git will not perform any conversions when checking out or committing text files. Choosing this option is not recommended for cross-platform projects ("core.autocrlf" is set to "false").


● Use MinTTY (the default terminal of MSYS2)

Git Bash will use MinTTY as terminal emulator, which sports a resizable window, non-rectangular selections and a Unicode font. Windows console programs (such as interactive Python) must be launched via `winpty` to work in MinTTY.

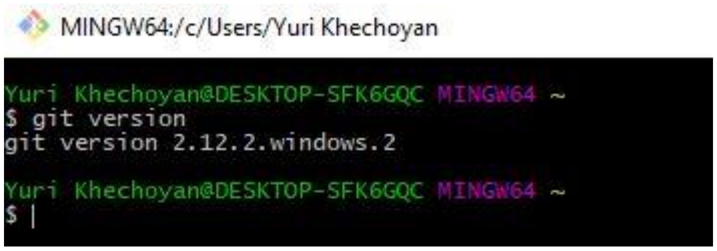
7. When choosing terminal emulator, select the 1st option: ***Use MinTTY (the default terminal of MSYS2)***
8. Continue clicking ***Next*** until installation starts

3.3 Initial Git Configuration

1. After installation is complete…

   *{Type in the git command: **git version**}*

   This will have Git print out onto the console, which version of Git is installed. This will not only tell you that Git has been successfully installed, but also a verification measurement to see if the version of Git installed matches the version downloaded. If they match, move to Step 2.

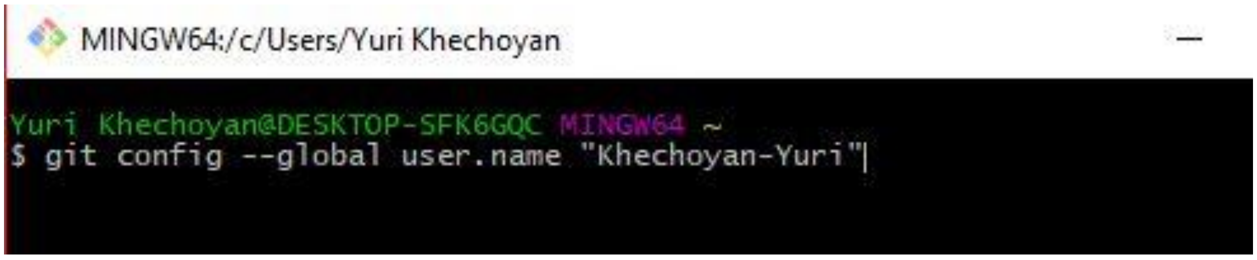


```
MINGW64:/c/Users/Yuri Khechoyan

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ git version
git version 2.12.2.windows.2

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ |
```

2. Next, we will need to have Git identified. This is done so that when you start making commits to GitHub, your computer can sign off on those changes and let GitHub know that it is you that is pushing these changes. In place of Username, type in your name.

   Note: Sections 3.3 and 3.4 are only done once during initial setup. You will not do this again unless you install git on another system.

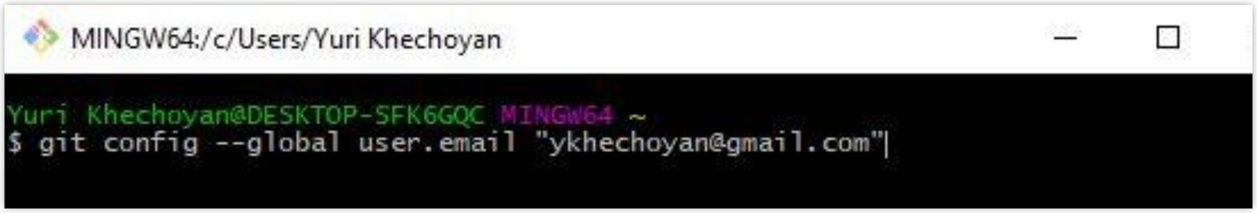
   *{Type in the command: **git config --global user.name "Username"**}*



```
MINGW64:/c/Users/Yuri Khechoyan

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ git config --global user.name "Khechoyan-Yuri"|
```

3. Then do the same thing for email. In place of email, type in your email (ex. name@domain.com)

{Type in the command: ***git config --global user.email*** name@domain.com}

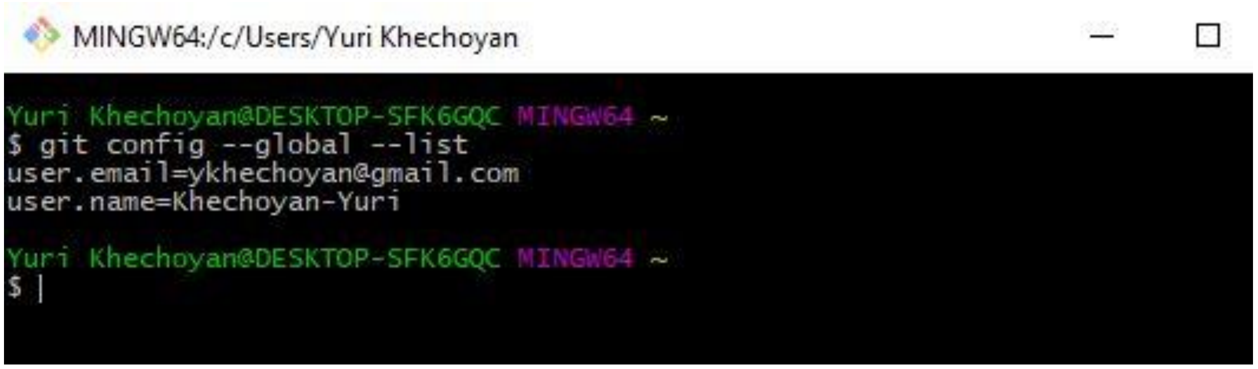


```
MINGW64:/c/Users/Yuri Khechoyan

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ git config --global user.email "ykhechoyan@gmail.com"
```

3.4 Initial Git Configuration Verification

4. To verify that the correct credentials have been entered…

{Type in the command: ***git config --global --list***}



```
MINGW64:/c/Users/Yuri Khechoyan

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ git config --global --list
user.email=ykhechoyan@gmail.com
user.name=Khechoyan-Yuri

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ |
```

Note: If the Username has been entered incorrectly, repeat Step 2. Repeat Step 4 to verify again.

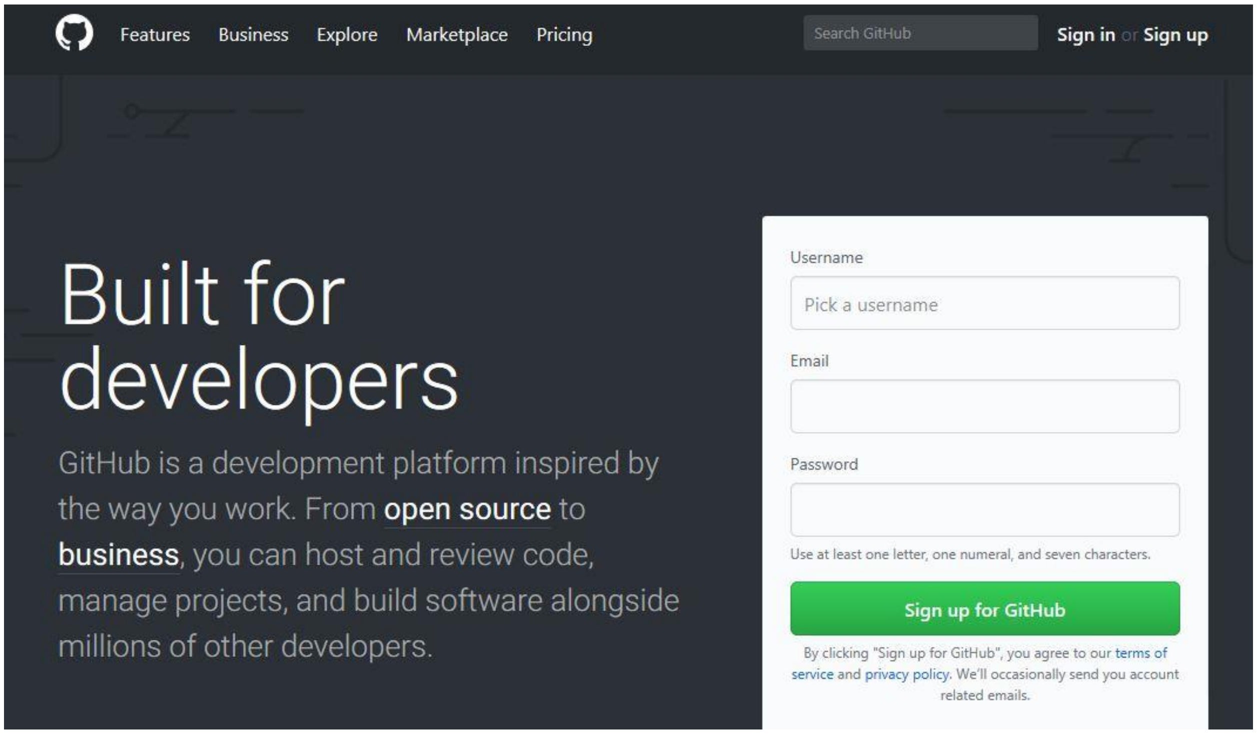
Note: If the email has been entered incorrectly, repeat Step 3. Repeat Step 4 to verify again.

4. Getting Started (GitHub)

4.1 Creating a GitHub Account

Congratulations! You have successfully downloaded and installed Git! Now onto the Cloud Based Repository.

1. Go to https://github.com/
2. Provide a Username, Password, and an Email
3. Click the green Sign Up Button
4. Verification email will be sent to email address you provided
5. Log In



Built for developers

GitHub is a development platform inspired by the way you work. From open source to business, you can host and review code, manage projects, and build software alongside millions of other developers.

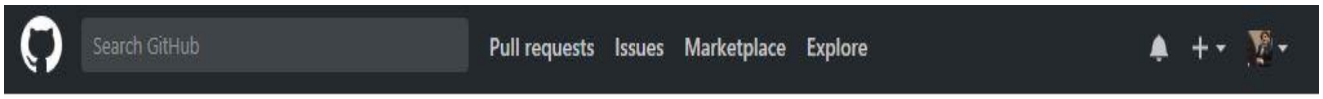
Use at least one letter, one numeral, and seven characters.

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

4.2 GitHub UI Overview

Once inside of the GitHub website, it may look overwhelming, but do not worry. It is easy to figure out what each part is and does.

At the top of the website is your navigation bar. This will be used to peruse through GitHub's site.



On the top left, is GitHub's Mascot appropriately named: ***Octocat!*** To the right of that is the search bar. The search bar can be used to search the entire GitHub database for:

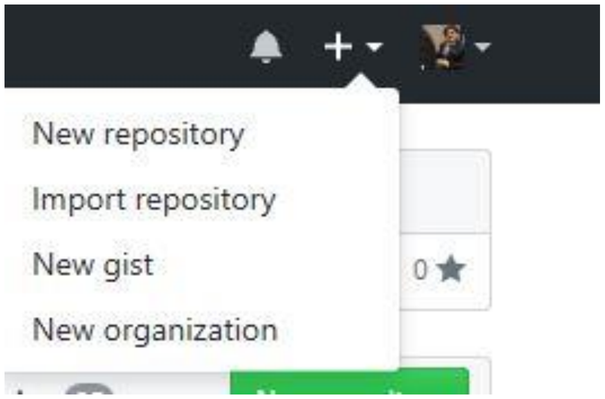
- Repositories by name (if you do not have the direct hyperlink to them)
- Users

To the right of the search bar are all your **Pull Requests**. Next to Pull Requests are your **Issues**. To the right of Issues is the **Marketplace**. This is where you integrate different 3rd party extension into your GitHub projects. To the right of Marketplace is **Explore**. Explore where you read up on what's new with GitHub, see trending repositories, see any upcoming GitHub events, and other resources.

Finally, on the far right hand side, you will find your **notifications**, the create button **"+"**, and **your account**.

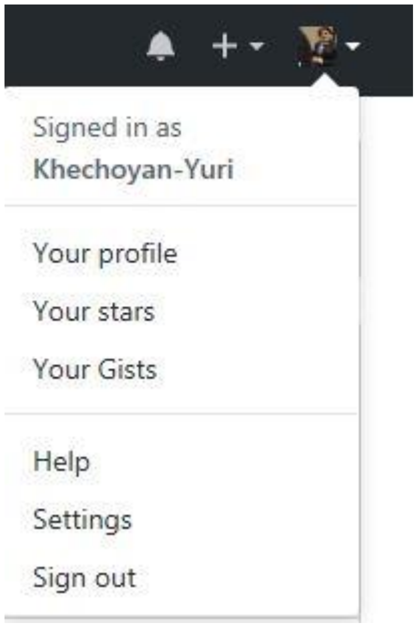
4.3 Nav. Bar, Far Right Side

The right-hand side of the Navigation Bar, is what you will use to navigate through the GitHub the most.



To the right of your Notifications is your create button. You will use this for a number of things:

- Create a new Repository (Private or Public)
- Import an existing Repository (your projects from GitHub's database)
- Create a new Gist (a text document)
- New Organization



When you click on your Profile picture, it opens up a different sub-menu.

- Your Profile
- Stars (Favorite Repositories)
- Gists
- Help
- Settings
- Sign Out

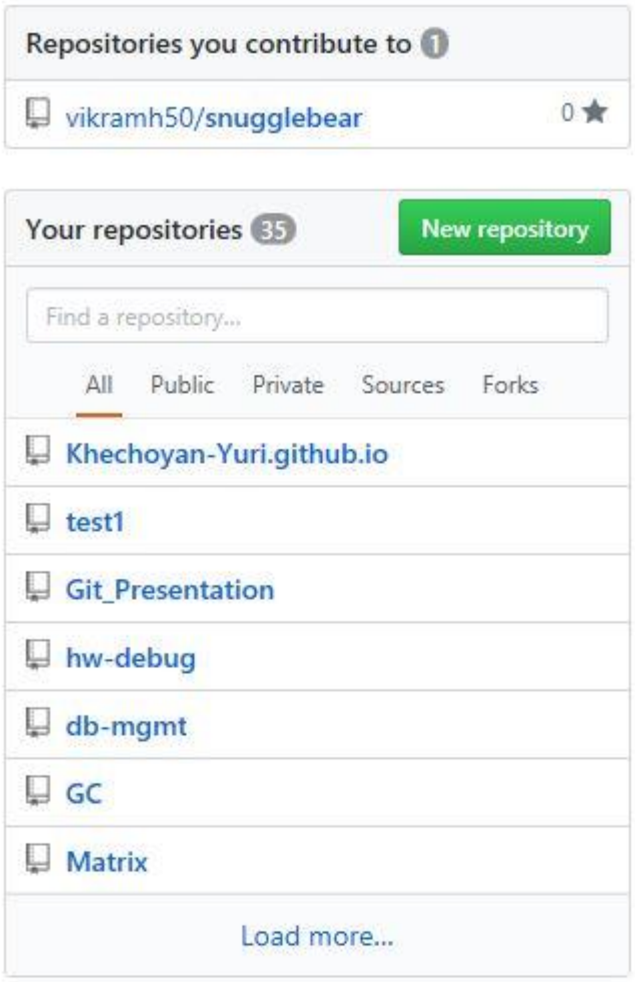
Settings:

Settings is where you will go to modify your information, add a payment method (if you plan to create Private Repositories), and Delete existing repositories that you have created.

4.4 Below Navigation Bar

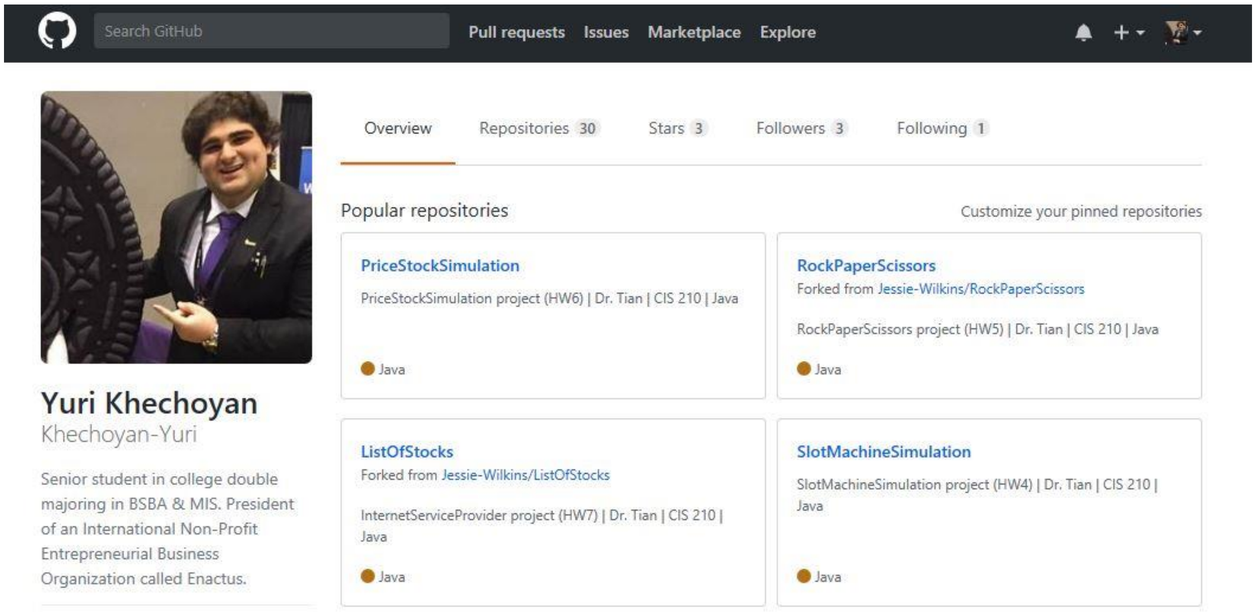
Below the Navigation Bar, you will find quick links to Repositories that you have contributed to and your personal repositories that you have created. Your repositories are categorized as follows:

- All
- Public
- Private (if you have a paid account)
- Sources (sources of your repositories)
- Forks (direct copies of other repositories that you have added to your own account)



4.5 Your Profile

Your profile consists of some very useful information.



Yuri Khechoyan
Khechoyan-Yuri

Senior student in college double majoring in BSBA & MIS. President of an International Non-Profit Entrepreneurial Business Organization called Enactus.

PriceStockSimulation
PriceStockSimulation project (HW6) | Dr. Tian | CIS 210 | Java

RockPaperScissors
Forked from Jessie-Wilkins/RockPaperScissors
RockPaperScissors project (HW5) | Dr. Tian | CIS 210 | Java

ListOfStocks
Forked from Jessie-Wilkins/ListOfStocks
InternetServiceProvider project (HW7) | Dr. Tian | CIS 210 | Java

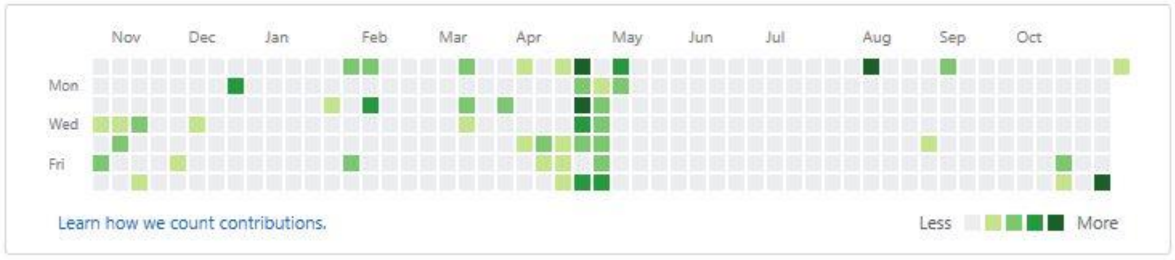
SlotMachineSimulation
SlotMachineSimulation project (HW4) | Dr. Tian | CIS 210 | Java

- Your profile picture
- A blurb “About You”
- Your popular repositories
- Repositories (All of the repositories you have created)
- Stars (Your favorite repositories)
- Followers (people that follow your work)
- Following (people that you are following)

If you scroll further down, you will find some more useful information...

173 contributions in the last year

Contribution settings ▾



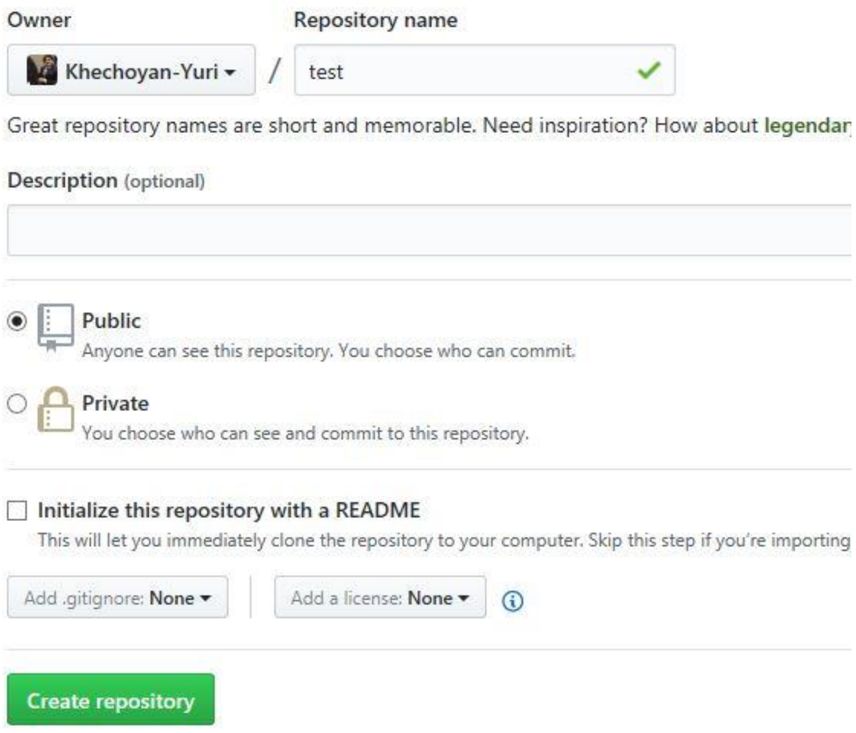
There is a timeline of all of the commits that you have made in the past year. The darker the green the squares are, the more commits you have made during that day. Each day of the year is denoted by a single square.

5. Creating Your First Repository (GitHub)

5.1 New Repository

When creating a new repository, there are a few things that require your input:

- Repository Name
- Description (can add this later)
- Public or Private Classification
- Initializing with a README.md
- .gitignore restrictions
- Licensing restrictions



Owner
Khechoyan-Yuri / Repository name
test ✓

Great repository names are short and memorable. Need inspiration? How about legendar

Description (optional)

● Public
Anyone can see this repository. You choose who can commit.

○ Private
You choose who can see and commit to this repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing

Add .gitignore: None
Add a license: None

When choosing a name for your repository, GitHub will automatically search through its database and see if there is a pre-existing repository within your own account that has the same name. If so, GitHub will tell you. Otherwise if the repository name has not been taken, you will see a green check mark next to the name.

If are wanting to create a private repository and have a method of payment on your account, choose Private. Otherwise, leave it on Public.

Note: Public repositories can be seen by the public but only the owner has the ability to commit and authorize other developers to make commits as well. Private repositories can only be seen and committed to by the owner and authorized developers.

Initializing with a README.md – just do it. It can be deleted later.

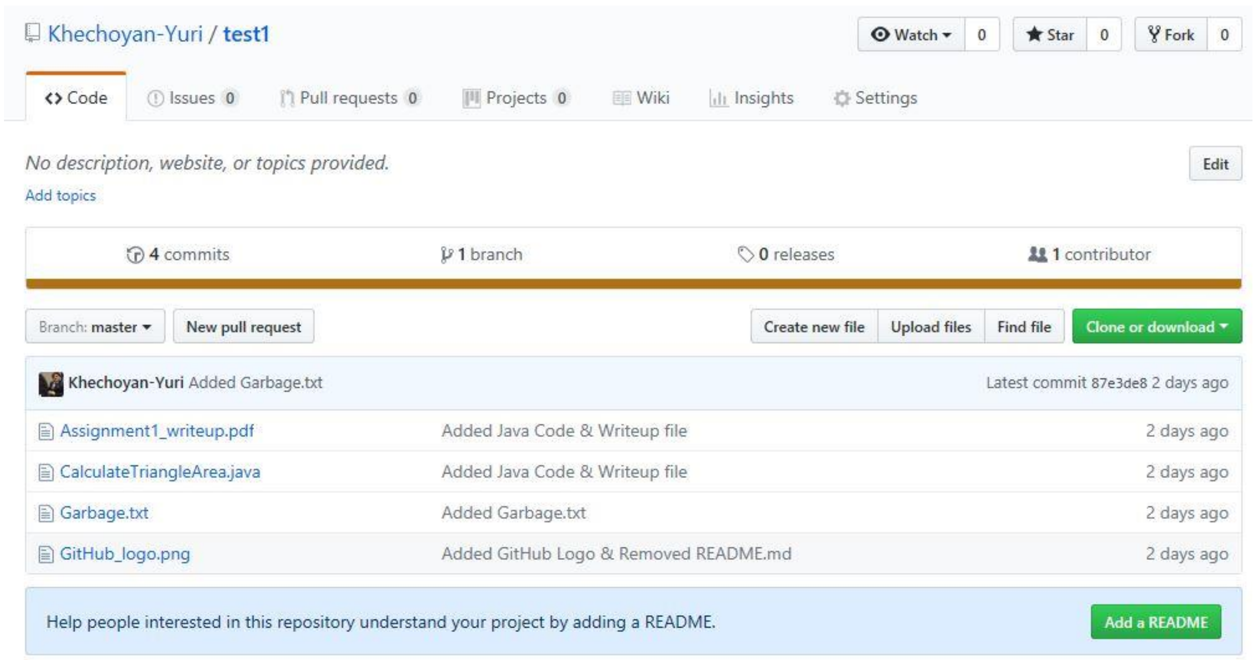
You can also customize the type of file you want git to ignore and incorporating the type of license you want your repository/project to have.

For now, leave both of them at their default values: none.

5.2 Your (Semi) First Commit

Congratulations! You have created your new repository! You did not see it, but when you created your repository, you actually made the first commit!

Once inside the main directory of your repository on GitHub, it will look something like this:



No description, website, or topics provided.

4 commits | 1 branch | 0 releases | 1 contributor

Khechoyan-Yuri Added Garbage.txt — Latest commit 87e3de8 2 days ago

Assignment1_writeup.pdf	Added Java Code & Writeup file	2 days ago
CalculateTriangleArea.java	Added Java Code & Writeup file	2 days ago
Garbage.txt	Added Garbage.txt	2 days ago
GitHub_logo.png	Added GitHub Logo & Removed README.md	2 days ago

Help people interested in this repository understand your project by adding a README.

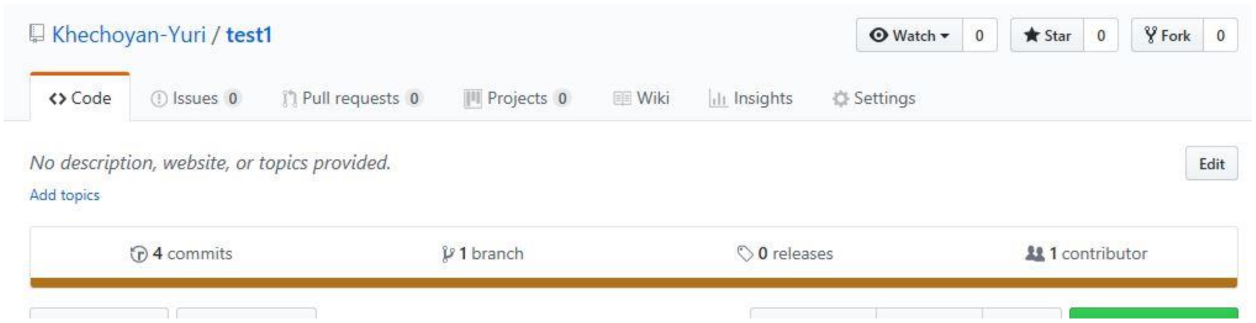
You: But Yuri, my directory looks a lot different than this. WHY?!

Me: I'm glad you asked beautiful/handsome stranger. Reason is because I added a few files and instead of a recorded 1 commit, I have 4. But in general, the UI is the same. When you click on the 4 **commits**, it will take you into an archive containing all of the commits that were made by you and your authorized contributors.

Branches are exactly what they sound like. Want to experiment with your code without affecting the final product, create a branch! The default branch that is always used is called the *master* branch. **Releases** are also pretty self-explanatory. Milestones in the code that you and your team is ready to release (v1.0, v2.36, v192.168.2.1, and so on….)

And **contributors** are all of the people that are working on this repository/project – whether that be you or other people authorized by you.

5.3 The Stars are Watching the Forks



No description, website, or topics provided.

At the top of your repository directory you will find some useful information as well.

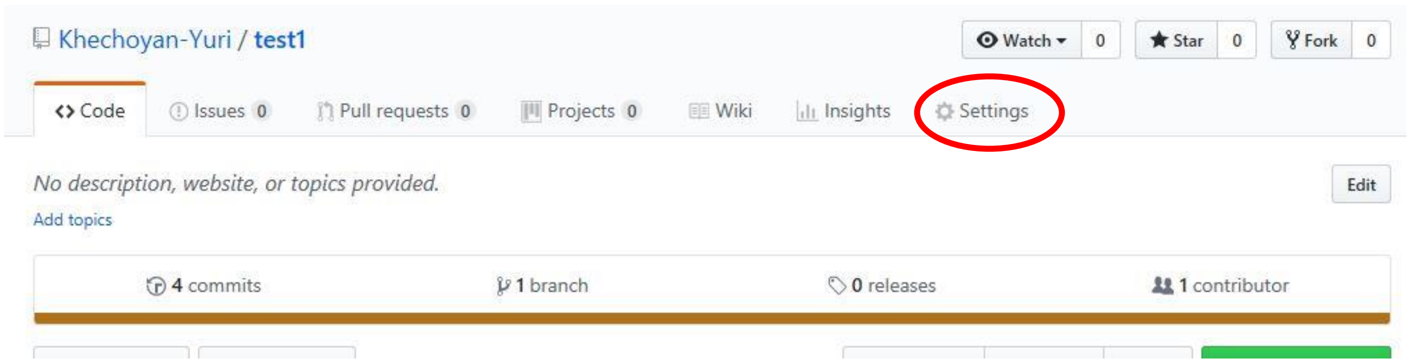
- Code (All of your files; where you will spend most of your time)
- Issues (any known and record issues within that project/Repo)
- Pull Requests (when someone makes changes and they want the changes to be reviewed before officially making a commit)
- Wiki (Wikipedia for your project/Repo)
- Insights (Analytics for the project/Repo)
- Settings (Renaming Repo, Adding Contributors, Deleting Repo, and more)

Then above that, you will find **Watch**, **Star**, and **Fork**. Watch means the number of users that are watching the project outside of the contributors list. Stars is the amount of times your project has been favorited. Forks are the number of users that have made a copy & paste of your entire repository on their accounts so that they can have their own version to play around with, improve on, etc.

5.4 Ocean’s 1011 (Creating a team)

When creating a team (adding contributors) to your newly created repository/project, there are things that need to occur:

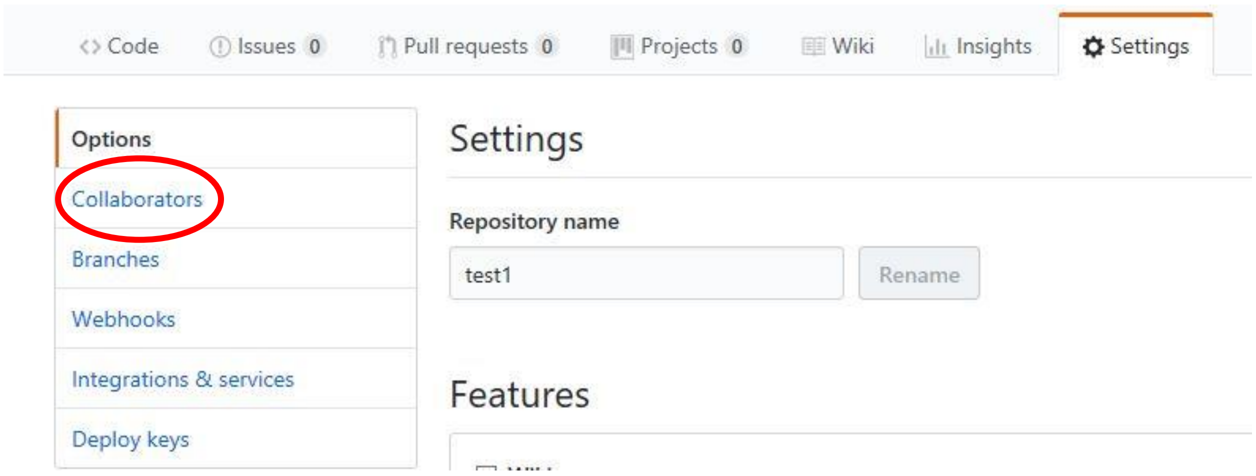
- You must authorize them
- They must accept your authorization invite



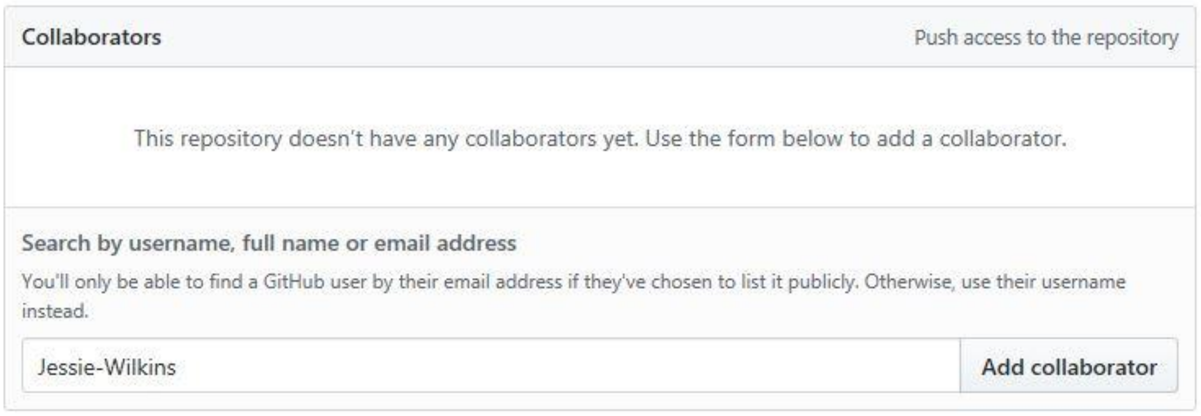
Since we are still in the main directory of

your repo on GitHub, click the on **Settings** tab.

Next, on the left-hand side you will see more sections that you could go into about that project. We will click on **Collaborators**.



Finally, you will search for your collaborator by their GitHub Username. If you do not know their GitHub username, no worries. GitHub gives you the ability to find them by email as well. The contributor will be added to the project when they accept your authorized invitation.



Collaborators

Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

Jessie-Wilkins

5.5 Local Repository Creation with Git

You’re still here? Alright, let’s continue. Now that you have your GitHub repo created, it’s time to go back to Git and create a local repo. When completed, we will then link the two repos together so that we could make commits later.

Since we have already completed the steps to configure your Git (back in section 3.3 & 3.4), we will not repeat this step.

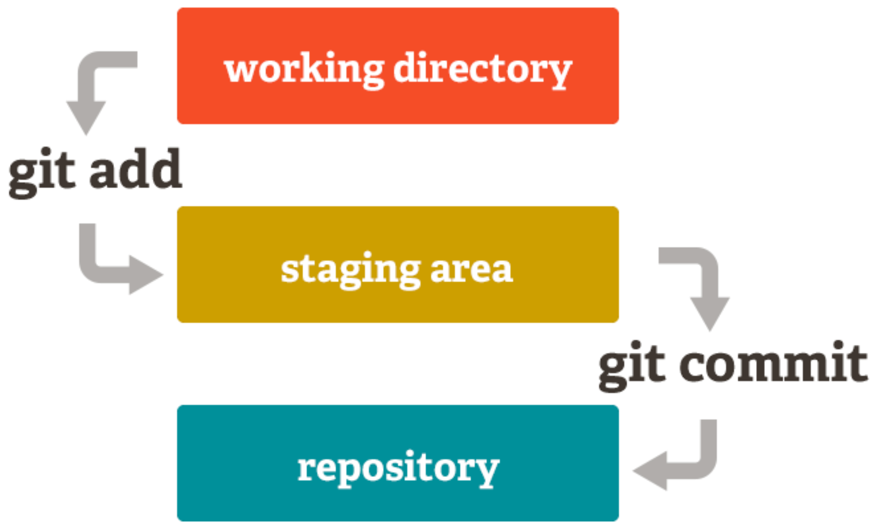
6. Back to Git

6.1 Commander in Chief (Git Commands)

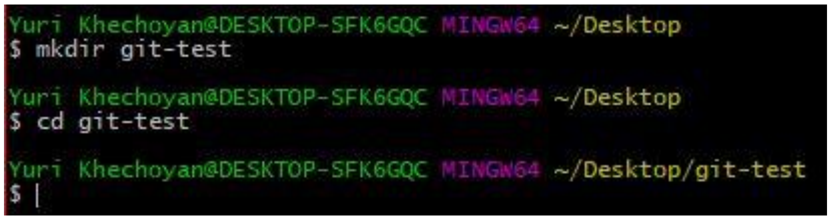
Now it’s time to get into the reason for this book’s existence.

Note: These commands are very powerful. Follow them in the order that they appear. If these commands are not typed in accurately, the action associated with each command will not be executed. On the contrary, it is possible to completely destroy your file system if you are not careful with these commands. Also, any time you see these brackets: <>, ignore them. They are just placeholders for you. They should not be entered as part of the commands.

Below is a diagram to illustrate how you control source code and other files within Git in order to upload from your computer (working directory) to GitHub (repository):

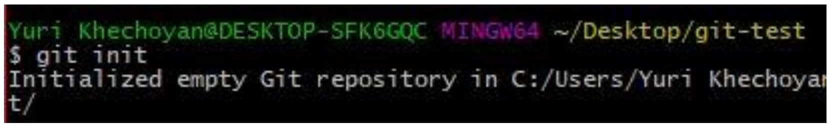


Git Commands (blue) & their meanings



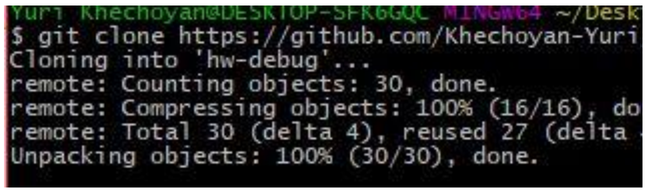
```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ mkdir git-test

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ cd git-test

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ |
```

mkdir <Folder name> - *mkdir* stands for Make Directory. Same as right clicking & creating a new folder **DO NOT include the "<>".** Those are just a place holder for us



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ git init
Initialized empty Git repository in C:/Users/Yuri Khechoyan
t/
```

git init – initializes the directory/folder that you are currently in to become a local Repository

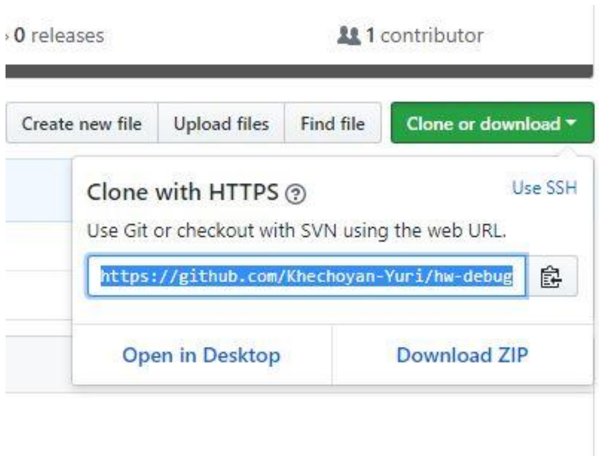


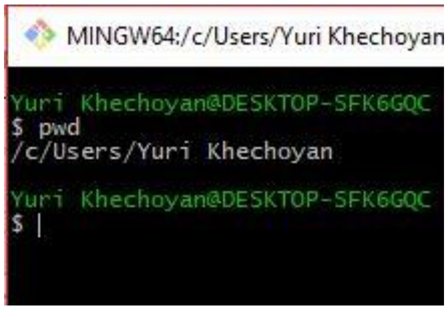
```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desk
$ git clone https://github.com/Khechoyan-Yuri
Cloning into 'hw-debug'...
remote: Counting objects: 30, done.
remote: Compressing objects: 100% (16/16), do
remote: Total 30 (delta 4), reused 27 (delta
Unpacking objects: 100% (30/30), done.
```

git clone <repository.git hyperlink> - used for the initial clone to have your local Repository match the one in GitHub. **DO NOT include the "<>".** Those are just a place holder for us. Hyperlink **<u>MUST</u>** have the .git extension. Clone command is executed, you will not use it until you create another local repo and match it to a different GitHub repo.

Obtaining the Repository Link

To obtain the repo link, on the main page of your project, there will be a green "Clone or Download" button. Click that and GitHub will show you which link you should use to clone, pull from, and push to.

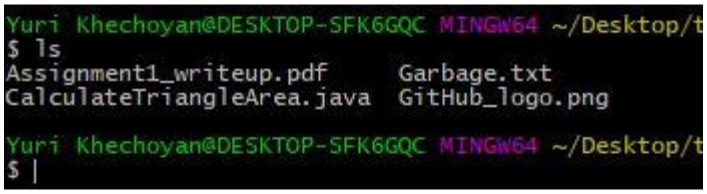




```
MINGW64:/c/Users/Yuri Khechoyan
Yuri Khechoyan@DESKTOP-SFK6GQC
$ pwd
/c/Users/Yuri Khechoyan

Yuri Khechoyan@DESKTOP-SFK6GQC
$ |
```

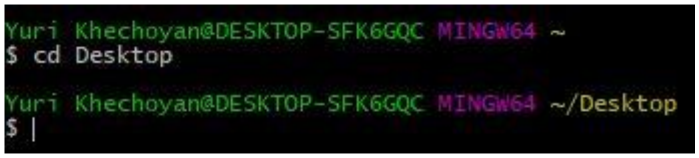
pwd – Stands for Print Working Directory. Tells you where on your computer Git is currently looking, (which Directory)

ls – (Lowercase L) Used to 'list' all contents in directory that Git is currently inside of..



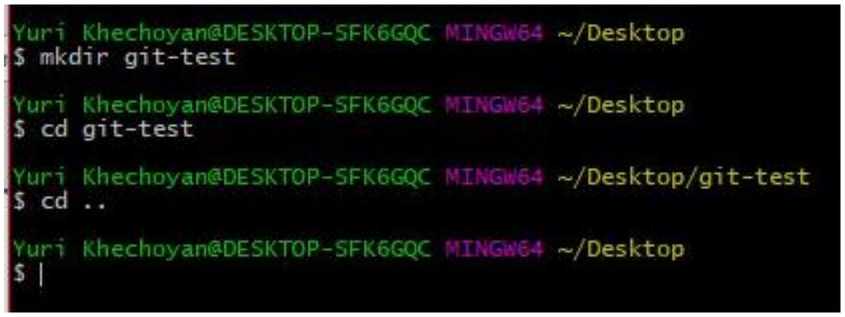
```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/t
$ ls
Assignment1_writeup.pdf     Garbage.txt
CalculateTriangleArea.java  GitHub_logo.png

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/t
$ |
```

(example: c/users/FirstName LastName/Desktop)



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~
$ cd Desktop

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ |
```

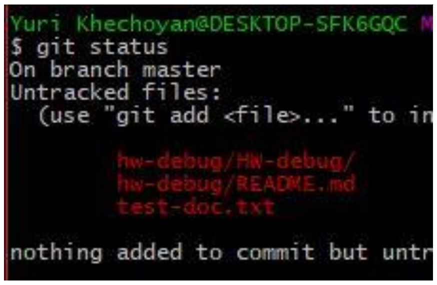
cd – Stands for Change Directory. Used to enter into and out of folders (like you would with mouse clicks) *cd ..*– (there is 1 space between the *d* and the first '.'): moves 1 directory backwards (example: if inside of FirstName/Documents, it will move into the FirstName/ directory).



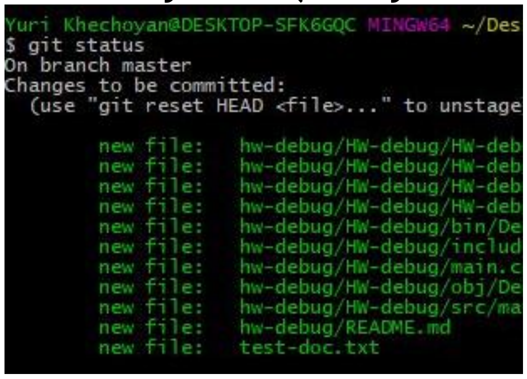
```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ mkdir git-test

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ cd git-test

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ cd ..

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop
$ |
```

git – any major action will start with 'git' in front of it

git status – gives you a status of your project. If files have been modified **BUT NOT** *add*ed to staging area by **YOU** (not your team), it will let you know by showing your contents in red. Otherwise if files have been modified **AND** *add*ed to staging area by **YOU** (not your team), contents will show up in green. When added, they are added to the staging area.



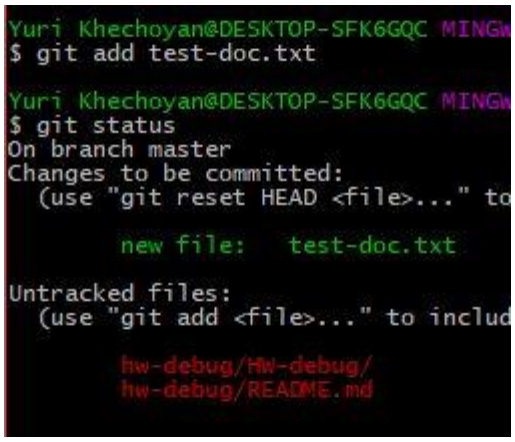
```
Yuri Khechoyan@DESKTOP-SFK6GQC M
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to in

        hw-debug/HW-debug/
        hw-debug/README.md
        test-doc.txt

nothing added to commit but untr
```



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Des
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage

        new file:   hw-debug/HW-debug/HW-deb
        new file:   hw-debug/HW-debug/HW-deb
        new file:   hw-debug/HW-debug/HW-deb
        new file:   hw-debug/HW-debug/HW-deb
        new file:   hw-debug/HW-debug/bin/De
        new file:   hw-debug/HW-debug/includ
        new file:   hw-debug/HW-debug/main.c
        new file:   hw-debug/HW-debug/obj/De
        new file:   hw-debug/HW-debug/src/ma
        new file:   hw-debug/README.md
        new file:   test-doc.txt
```

exit – exit does the same as the 'X' button in any program. It closes them. This is just the keyboard version for Git.

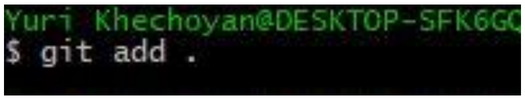
The Next few commands need to be done IN THIS ORDER TO UPLOAD (add, commit, push)



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW
$ git add test-doc.txt

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW
$ git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to

        new file:   test-doc.txt

Untracked files:
  (use "git add <file>..." to includ

        hw-debug/Hw-debug/
        hw-debug/README.md
```

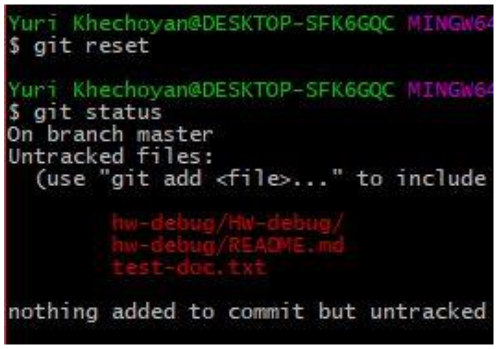
git add <insert file name> – adds whatever file to "staging area": the step before committing and uploading to repository. **When adding, DO NOT include the "<>".** Those are just a place holder for us. Make sure to include full name of file and extension.

---------------------OR YOU CAN USE----------------------------



```
Yuri Khechoyan@DESKTOP-SFK6GQ
$ git add .
```

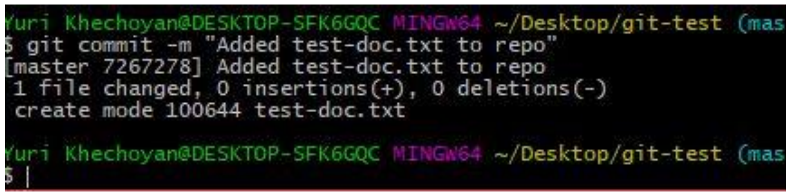
git add .– the '.' here, 1 space after "add" means *'add ALL files'*. Use 'add' for specific file modification or if you have revised multiple files, use the period instead (it's easier)



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64
$ git reset

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include

        hw-debug/Hw-debug/
        hw-debug/README.md
        test-doc.txt

nothing added to commit but untracked
```

git reset – In the case where files were wrongfully added to the staging area, use this command to undo the *git add* command

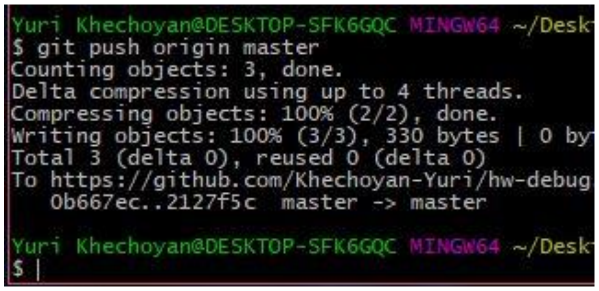
git commit -m "Add your commit message here" – commit lets Git know that at this time, this is the version you want to push to GitHub. There is a space between each major part (git, commit, -m, and the quotations). The *-m* is the message that you want accompanied with your commit. Inside the quotations is where



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test (mas
$ git commit -m "Added test-doc.txt to repo"
[master 7267278] Added test-doc.txt to repo
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 test-doc.txt

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test (mas
$ |
```

you would want to tell the people you are working with what you did: i.e. added something, revised something, deleted something, fixed bugs, etc. **And keep the message short! No more than 1 sentence!!!**

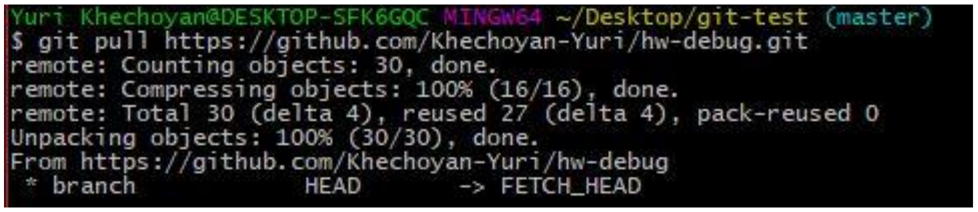
git push origin master – this is the final step needed in order to complete the upload. This command is actually needed in order to fully upload to GitHub. What this means is: I am pushing new files/code to the origin of master branch within my GitHub repository.



```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desk
$ git push origin master
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 330 bytes | 0 by
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Khechoyan-Yuri/hw-debug
   0b667ec..2127f5c  master -> master

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desk
$ |
```

When other people make changes and you want an updated version

git pull <repository.git link> this will be used to pull any new updates from the GitHub repository to your local Repo. Once you type in pull, right click with the mouse and paste the link (found below) then hit the enter key. **DO NOT include the "<>".** Those are just a place holder for us. Hyperlink **MUST** have the .git extension.



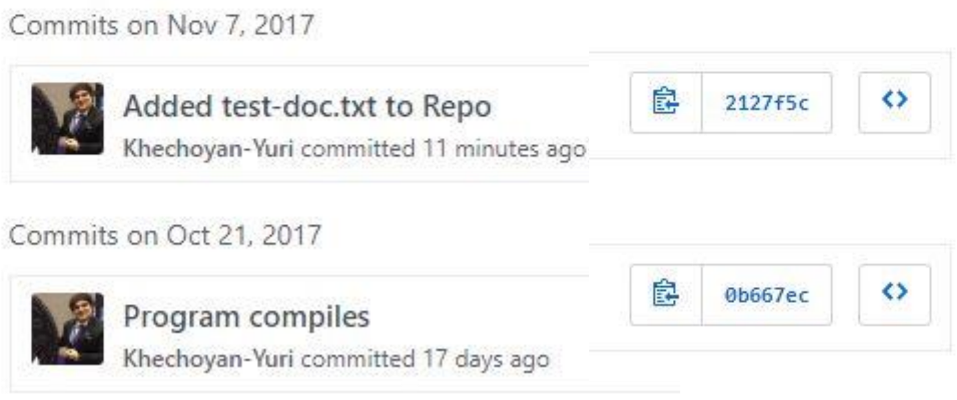
```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test (master)
$ git pull https://github.com/Khechoyan-Yuri/hw-debug.git
remote: Counting objects: 30, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 30 (delta 4), reused 27 (delta 4), pack-reused 0
Unpacking objects: 100% (30/30), done.
From https://github.com/Khechoyan-Yuri/hw-debug
 * branch            HEAD       -> FETCH_HEAD
```

Instructions on how to delete the latest commit and revert back to the older version

git reset <commit # reverting back to> --hard

git push origin -f

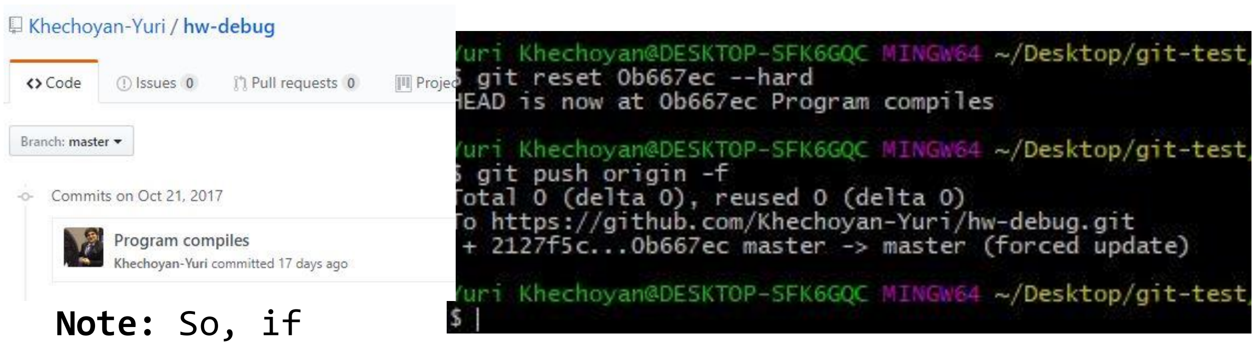
Even though you can see all of the commits made separately, this command is powerful and dangerous all at the same time. Make sure that if you decide to use this, you use it responsibly. This code will delete any and ALL commits to repo up to a certain commit. That is why you have to provide the commit # (7-character id). The latest version of the files is what is known as the "head". Think of it as the first person in line. When first person leaves, there is a new line leader. Every time you make a new commit, that specific commit becomes the head. And every time you execute this command, the previous commit behind it becomes the new head.


Commits on Nov 7, 2017

Added test-doc.txt to Repo
Khechoyan-Yuri committed 11 minutes ago
2127f5c

Commits on Oct 21, 2017

Program compiles
Khechoyan-Yuri committed 17 days ago
0b667ec


Khechoyan-Yuri / hw-debug

Commits on Oct 21, 2017

Program compiles
Khechoyan-Yuri committed 17 days ago

```
Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ git reset 0b667ec --hard
HEAD is now at 0b667ec Program compiles

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ git push origin -f
Total 0 (delta 0), reused 0 (delta 0)
To https://github.com/Khechoyan-Yuri/hw-debug.git
 + 2127f5c...0b667ec master -> master (forced update)

Yuri Khechoyan@DESKTOP-SFK6GQC MINGW64 ~/Desktop/git-test
$ |
```

Note: So, if let's say for an example, I made a mistake when uploading the latest commit on Nov. 7th because I later found out something went terribly wrong in the code, what I would do is execute: *git reset 0b667ec --hard*. Followed by the execution: *git push origin -f*. But if the mistake is not that drastic, you can override it by updating the code and making another commit.

So, what I am telling Git is: I made a mistake when making that commit, please completely delete it and go back to the version *0b667ec* of the code. Then I would like you to force push/override (*git push origin -f*) that change. Keep in mind, if you have Git go back further with your commits (for an example): go back to a commit made a year ago, that means that **any and all progress made with commits after that point will be deleted and you will not be able to retrieve them!**

As you can see (above), the commit made on Nov. 7th is longer in existence. It has been successfully removed.

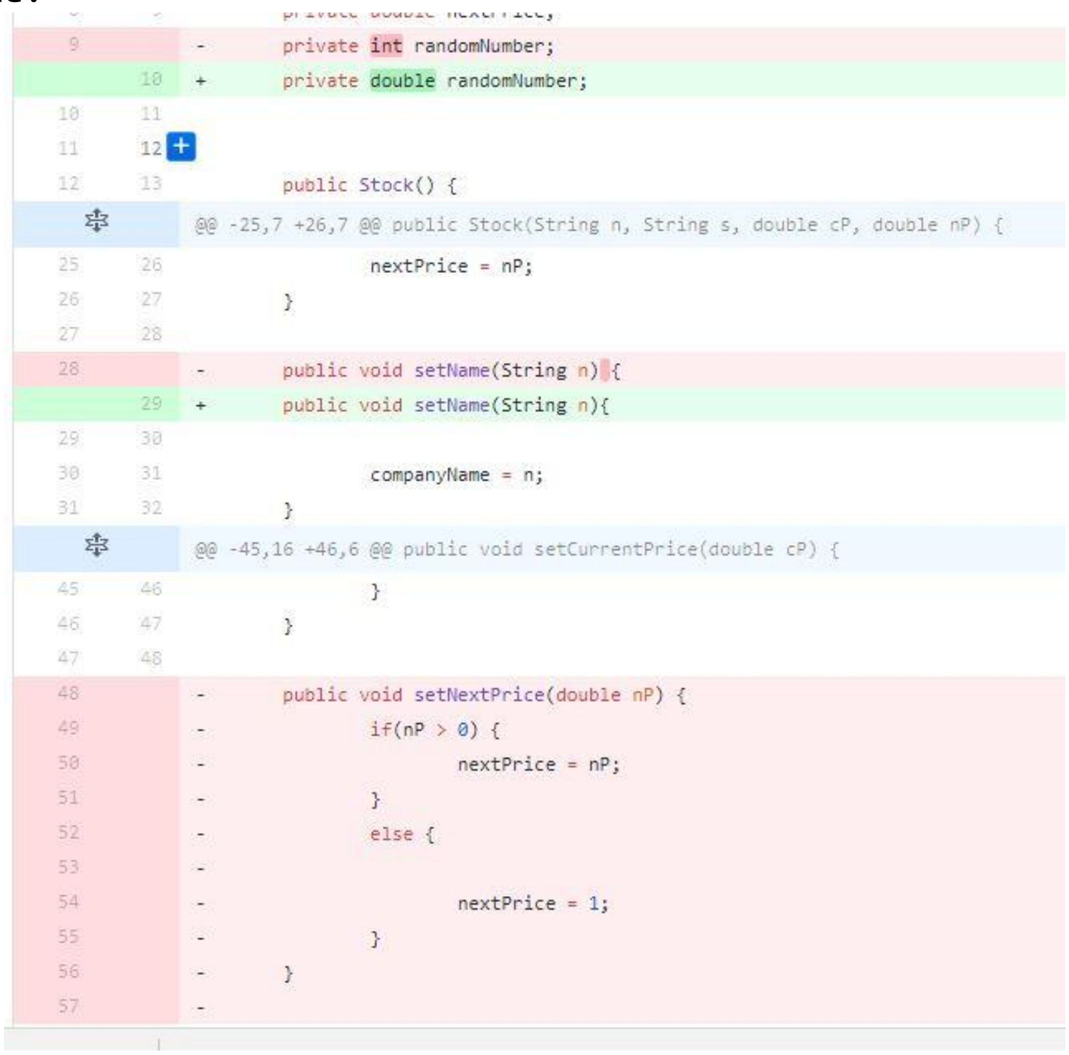
6.2 General Git Procedure

1. Type in: pwd so you know where Git is currently looking in
2. *cd* into wherever you would like to store the repo folder
3. Use *mkdir <Folder name>* if you haven't already created a folder to store your repo.
4. *cd* into the Repo folder that you have created in step 2.
5. Type in *git init* to initialize it.
6. Then use the *git clone* command (clone generally would be used at the very beginning. Then pull would be used every other time)
7. Once you have made any changes to the project, use the codes in this order (see above for exact code list):
   a. *git add*
   b. *git commit -m "message"*
   c. *git push origin master*
   d. *git pull* (when new changes have been made by other team members)

7. Analytics

7.1 Crunching Numbers and Crushing Bugs (Code Analytics)

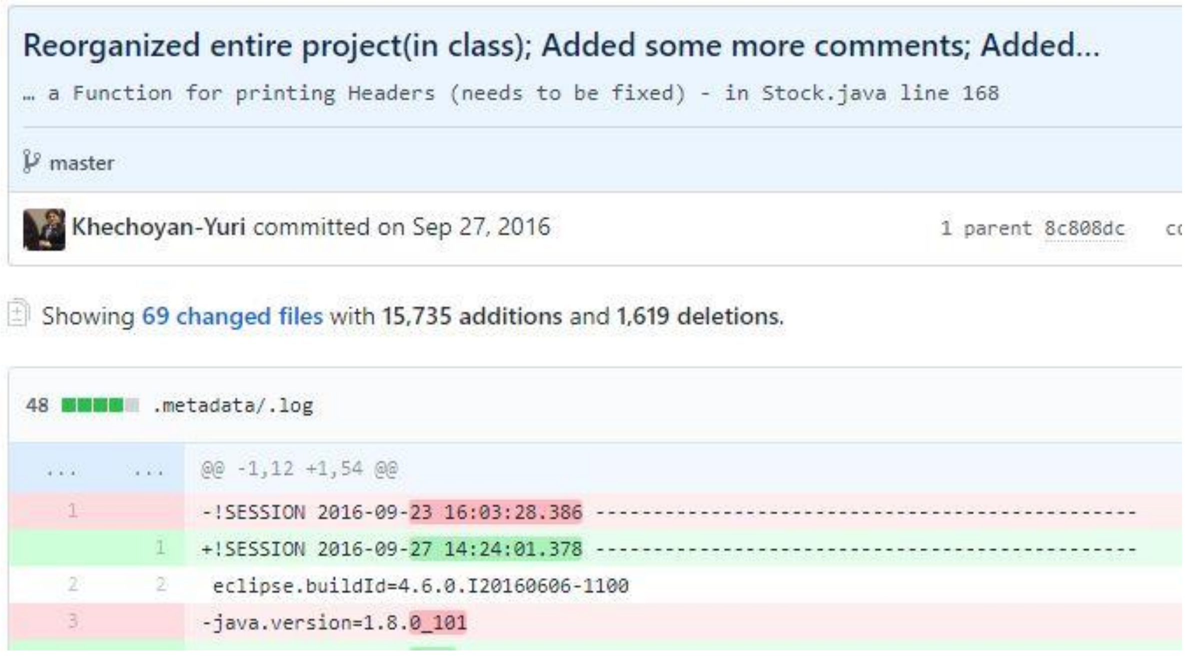
When working with and managing code, it is imperative to understand what has been done to the code in previous iterations (commits). When looking at the code after new commits have been pushed, everything that has been highlighted in green is all of the new code that has been added as part of that commit at that point in time. And subsequently, any code highlighted in red is all of the code that has been deleted for that specific commit at that given time.



```
-       private int randomNumber;
+       private double randomNumber;


        public Stock() {
@@ -25,7 +26,7 @@ public Stock(String n, String s, double cP, double nP) {
                nextPrice = nP;
        }

-       public void setName(String n) {
+       public void setName(String n){

                companyName = n;
        }
@@ -45,16 +46,6 @@ public void setCurrentPrice(double cP) {
                }
        }

-       public void setNextPrice(double nP) {
-               if(nP > 0) {
-                       nextPrice = nP;
-               }
-               else {
-
-                       nextPrice = 1;
-               }
-       }
-
```

7.2 Git Commit Summary

Git will even summarize the events that occurred when you jump into the exact commit number. As you can see from the image, for a certain commit that I have made on Sept. 27, 2016 – I have made 15,735 additions and 1,619 deletions. The reason why I have modified so much is because the commit message reads: “Reorganized entire project.” That is the main thing that I did in this commit.



Reorganized entire project(in class); Added some more comments; Added…

… a Function for printing Headers (needs to be fixed) - in Stock.java line 168

master

Khechoyan-Yuri committed on Sep 27, 2016 1 parent 8c808dc

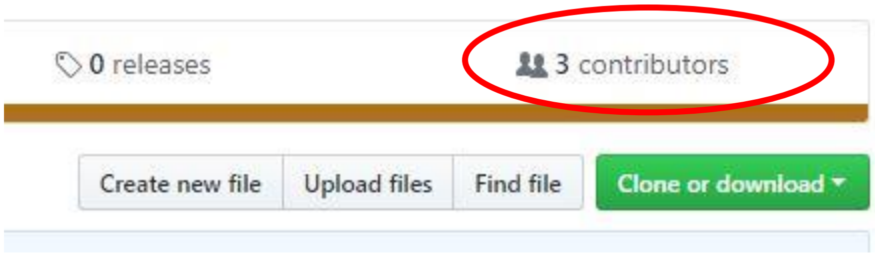
Showing 69 changed files with 15,735 additions and 1,619 deletions.

48 .metadata/.log

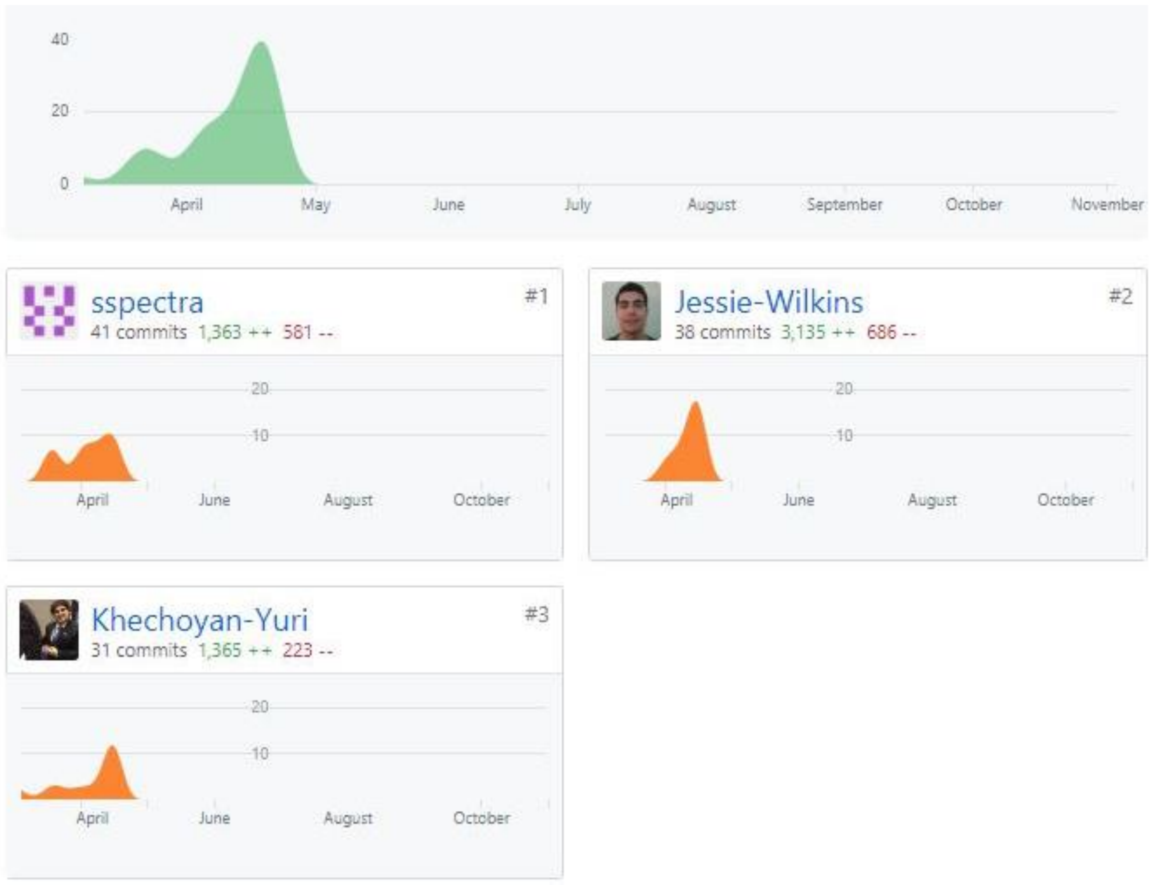
```
...    ...    @@ -1,12 +1,54 @@
1             -!SESSION 2016-09-23 16:03:28.386 ---------------------------------------------
       1      +!SESSION 2016-09-27 14:24:01.378 ---------------------------------------------
2      2       eclipse.buildId=4.6.0.I20160606-1100
3             -java.version=1.8.0_101
```

7.3 Contributions Summary

When you want to see who has had the most amount of activity within a project, click on the **Contributors** link from the main directory of your repo.



Once accessed, you can see who has had the most activity within the project/repo.



sspectra
41 commits 1,363 ++ 581 --

Jessie-Wilkins
38 commits 3,135 ++ 686 --

Khechoyan-Yuri
31 commits 1,365 ++ 223 --

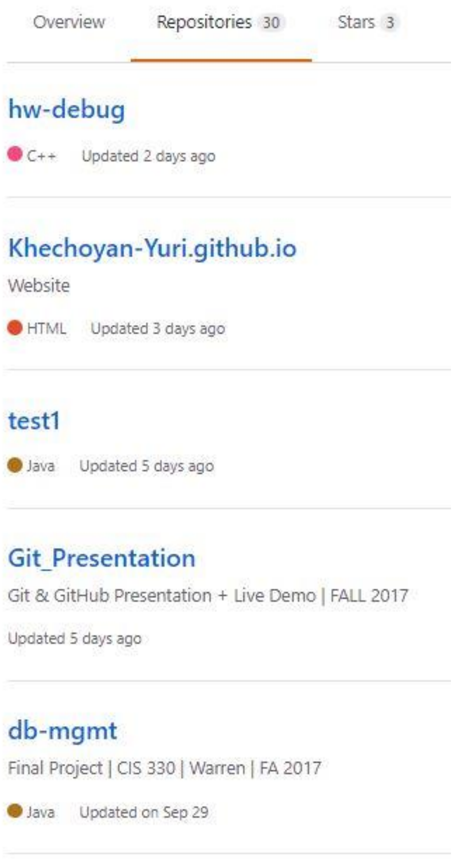
7.4 What’s Your Programming Poison? (language colors)

No matter what language you program in, GitHub has a way of identifying your source code language by the extension of those files. On GitHub, each language has a unique color identifier associated with it.

To see the full list of colors: Click Here

There are several ways for GitHub to let you know that it has identified your coding language(s) within projects.

Option 1: is in your profile under Repositories.



hw-debug

● C++ Updated 2 days ago

Khechoyan-Yuri.github.io

Website

● HTML Updated 3 days ago

test1

● Java Updated 5 days ago

Git_Presentation

Git & GitHub Presentation + Live Demo | FALL 2017

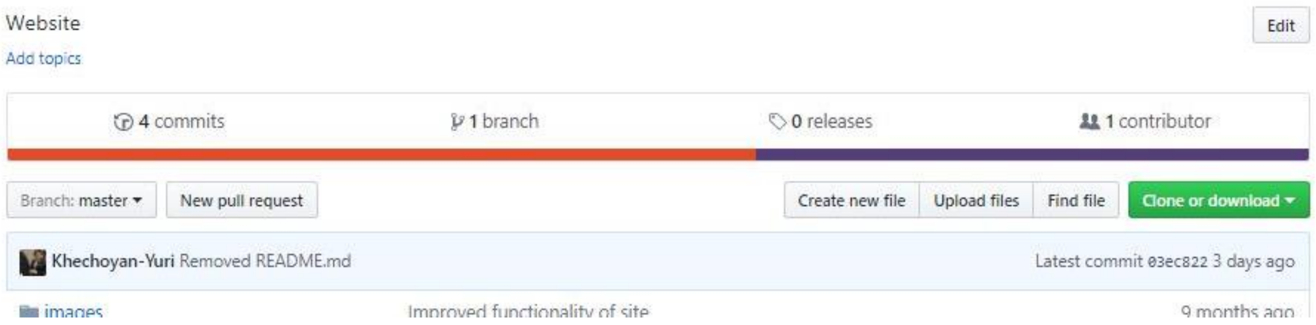
Updated 5 days ago

db-mgmt

Final Project | CIS 330 | Warren | FA 2017

● Java Updated on Sep 29

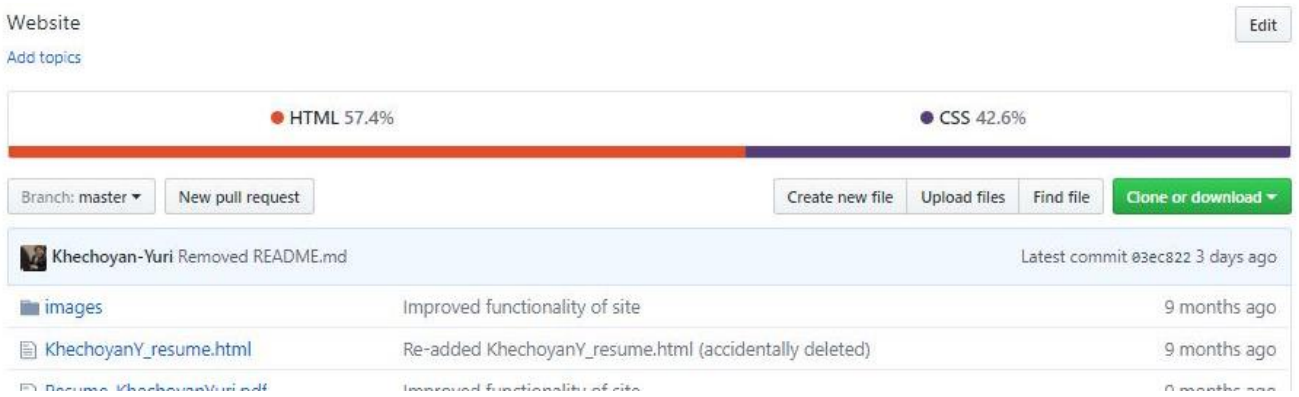
Option 2: If you are inside the main directory of your project, the bar going across underneath your commits, branches, releases, and contributors is the color associated with the languages used within your project.



Khechoyan-Yuri Removed README.md   Latest commit 03ec822 3 days ago

images   Improved functionality of site   9 months ago

If you click anywhere on that bar, the bar will execute an animation, revealing what languages were used in the project. And as an added bonus, it will even show you what percentage of the project has been completed in that language!



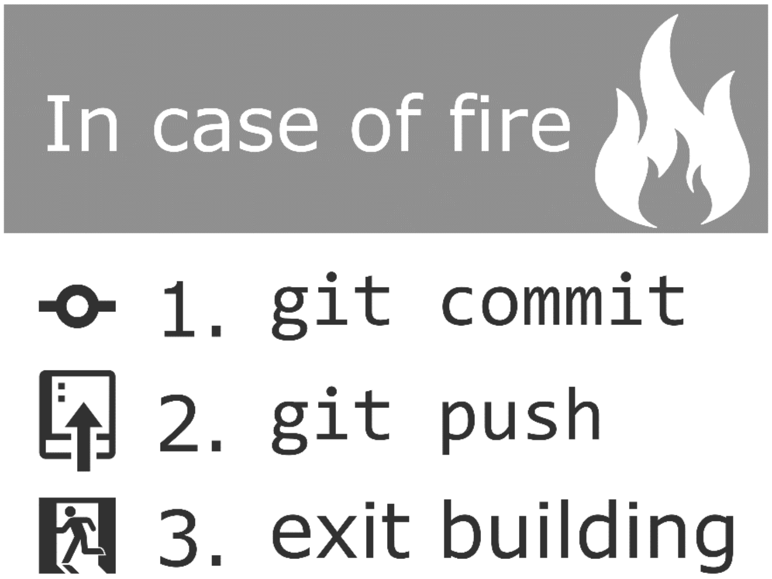
Khechoyan-Yuri Removed README.md   Latest commit 03ec822 3 days ago

images	Improved functionality of site	9 months ago
KhechoyanY_resume.html	Re-added KhechoyanY_resume.html (accidentally deleted)	9 months ago

8. End Notes

8.1 Fire Hazard Tips

When working with code, if there is ever a fire, remember...



8.2 Copyright

Copyright Act, 17 U.S.C. § 101 Definitions:

Except as otherwise provided in this title, as used in this title, the following terms and their variant forms mean the following:

A “computer program” is a set of statements or instructions to be used directly or indirectly in a computer in order to bring about a certain result.
“Copies” are material objects, other than phonorecords, in which a work is fixed by any method now known or later developed, and from which the work can be perceived, reproduced, or otherwise communicated, either directly or with the aid of a machine or device. The term “copies” includes the material object, other than a phonorecord, in which the work is first fixed.
“Copyright owner”, with respect to any one of the exclusive rights comprised in a copyright, refers to the owner of that particular right.

A “device”, “machine”, or “process” is one now known or later developed.

A work is “fixed” in a tangible medium of expression when its embodiment in a copy or phonorecord, by or under the authority of the author, is sufficiently permanent or stable to permit it to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration. A work consisting of sounds, images, or both, that are being transmitted, is “fixed” for purposes of this title if a fixation of the work is being made simultaneously with its transmission.

An “international agreement” is:
(1) the Universal Copyright Convention;
(2) the Geneva Phonograms Convention;
(3) the Berne Convention;
(4) the WTO Agreement;
(5) the WIPO Copyright Treaty;
(6) the WIPO Performances and Phonograms Treaty; and
(7) any other copyright treaty to which the United States is a party.

A “joint work” is a work prepared by two or more authors with the intention that their contributions be merged into inseparable or interdependent parts of a unitary whole.

For purposes of section 513, a “proprietor” is an individual, corporation, partnership, or other entity, as the case may be, that owns an establishment or a food service or drinking establishment, except that no owner or operator of a radio or television station licensed by the Federal Communications Commission, cable system or satellite carrier, cable or satellite carrier service or programmer, provider of online services or network access or the operator of facilities therefor, telecommunications company, or any other such audio or audiovisual service or programmer now known or as may be developed in the future, commercial subscription music service, or owner or operator of any other transmission service, shall under any circumstances be deemed to be a proprietor.

“Publication” is the distribution of copies or phonorecords of a work to the public by sale or
other transfer of ownership, or by rental, lease, or lending. The offering to distribute copies
or phonorecords to a group of persons for purposes of further distribution, public performance,
or public display, constitutes publication. A public performance or display of a work does not of
itself constitute publication.

To perform or display a work “publicly” means:
(1) to perform or display it at a place open to the public or at any place where a substantial
number of persons outside of a normal circle of a family and its social acquaintances is gathered; or
(2) to transmit or otherwise communicate a performance or display of the work to a place specified by
clause (1) or to the public, by means of any device or process, whether the members of the public capable of receiving the performance or display receive it in the same place or in separate places and at the
same time or at different times.

“Registration”, for purposes of sections 205(c)(2), 405, 406, 410(d), 411, 412, and 506(e),
means a registration of a claim in the original or the renewed and extended term of copyright.
“Sound recordings” are works that result from the fixation of a series of musical, spoken, or other
sounds, but not including the sounds accompanying a motion picture or other audiovisual work, regardless
of the nature of the material objects, such as disks, tapes, or other phonorecords, in which they are embodied.
“State” includes the District of Columbia and the Commonwealth of Puerto Rico, and any territories to
which this title is made applicable by an Act of Congress.

A “transfer of copyright ownership” is an assignment, mortgage, exclusive license, or any other conveyance, alienation, or hypothecation of a copyright or of any of the exclusive rights comprised in a copyright,
whether or not it is limited in time or place of effect, but not including a nonexclusive license.

A “transmission program” is a body of material that, as an aggregate, has been produced for the sole
purpose of transmission to the public in sequence and as a unit.

To “transmit” a performance or display is to communicate it by any device or process whereby images or
sounds are received beyond the place from which they are sent.

For purposes of section 411, a work is a “United States work” only if:

(1) in the case of a published work, the work is first published—
(A) in the United States;
(B) simultaneously in the United States and another treaty party or parties, whose law grants a
term of copyright protection that is the same as or longer than the term provided in the United States;
(C) simultaneously in the United States and a foreign nation that is not a treaty party; or

(D) in a foreign nation that is not a treaty party, and all of the authors of the work are nationals, domiciliaries, or habitual residents of, or in the case of an audiovisual work legal entities with headquarters in, the United States;

(2) in the case of an unpublished work, all the authors of the work are nationals, domiciliaries, or habitual residents of the United States, or, in the case of an unpublished audiovisual work, all the authors are legal entities with headquarters in the United States; or

(3) in the case of a pictorial, graphic, or sculptural work incorporated in a building or structure, the building or structure is located in the United States.
A “useful article” is an article having an intrinsic utilitarian function that is not merely to portray the appearance of the article or to convey information. An article that is normally a part of a useful article is considered a “useful article”.

The author’s “widow” or “widower” is the author’s surviving spouse under the law of the author’s domicile at the time of his or her death, whether or not the spouse has later remarried.
The “WIPO Copyright Treaty” is the WIPO Copyright Treaty concluded at Geneva, Switzerland, on December 20, 1996.
The “WIPO Performances and Phonograms Treaty” is the WIPO Performances and Phonograms Treaty concluded at Geneva, Switzerland, on December 20, 1996.

Intellectual Property and Communications Omnibus Reform Act of 1999, as enacted by section 1000(a)(9) of Public Law 106-113, nor the deletion of the words added by that amendment:

(A) shall be considered or otherwise given any legal significance, or

(B) shall be interpreted to indicate congressional approval
or disapproval of, or acquiescence in,
any judicial determination,
by the courts or the Copyright Office. Paragraph (2) shall
be interpreted as if both section 2(a)(1)
of the Work Made for Hire and Copyright Corrections Act of
2000 and section 1011(d) of the Intellectual
Property and Communications Omnibus Reform Act of 1999, as
enacted by section 1000(a)(9) of Public Law 106-113,
were never enacted, and without regard to any inaction or
awareness by the Congress at any time of any judicial
determinations.