Calculator

Setup

This exercise will have you implementing a prefix calculator CLI application (an application that runs in the terminal). The calculator lets a user do things like add, subtract, multiply, and divide numbers.

Download the materials for this exercise clicking the Download link on Frodo.

Once the materials have finished downloading, open your command line environment and ***cd*** into the project directory.

Then, initialize a Git repo, ***git add*** all relevant files, and make an initial commit:

```
$ git init
$ git add calculator.js
$ git commit -am "Initial commit"
```

Get in the habit of doing this at the beginning of every exercise. You don’t want to be in a situation where you’re knee-deep in an exercise but you’ve forgotten to use Git and then your pet eats your computer, causing you to lose all your work! Also, it’s nice having that initial commit to record what your project files looked like when you first recieved them.

Introduction

Calculators and Notation

We typically write math equations using ***infix notation*** like this — `3 + 2` — where the operator (in this case, the plus sign) goes *inside* the numbers.

Another way to write equations is with ***prefix notation***. As the name implies, in prefix notation, operators go *before* the operands so, instead of `3 + 2`, you'd write `+ 3 2`.

One advantage of prefix notation is the ability to have an arbitrary number of operands. For example, `4 + 3 + 2` in prefix notation is `+ 4 3 2`.

In this exercise, we will build a basic prefix calculator together. We will provide half of the program, and you will code the other half. We will only handle 2 operands for the calculator in this exercise.

Your Task

Implement a prefix notation calcuator in the file called ***calculator.js***. You need to handle the following mathematical operations:

- Addition
- Subtraction
- Multiplication
- Division
- Square root

We've provided some code that allows you to access the numbers and the math symbol that the user provided on the command line. Review the code carefully in ***calculator.js***. Research any syntax that looks unfamiliar to you.

In order to see what the code is doing, add a ***console.log*** statement that prints out each variable in your calculator function. It's helpful to add a "label" in your console.log, with the variable's name, like so:

```
console.log('mathSymbol', mathSymbol);
```

Hover below if you need help.

Hint: Printing Variables Out

▼ *Click to hide*

```
reader.question("What would you like to calculate?", function(input){
	tokens = input.split(' ');
```

```
            mathSymbol = tokens[0];
            num1 = Number(tokens[1]);
            num2 = Number(tokens[2]);

            console.log('mathSymbol', mathSymbol);
            console.log('num1', num1);
            console.log('num2', num2);

            reader.close()

    });
```

Now, run the file to see your output. **You need to be in the same directory as the* `calc.js` *file in order to run the command below.**

```
$ node calc.js
```

Next: Handling Addition

Below your ***console.log*** statements, write an if-statement that checks to see if the ***mathSymbol*** variable is equal to the string `"+"`.

Hover below if you need help.

Hint: Checking for Addition Symbol

▼ *Click to hide*

```
reader.question("What would you like to calculate?", function(input){
      tokens = input.split(' ');

  mathSymbol = tokens[0];
      num1 = Number(tokens[1]);
      num2 = Number(tokens[2]);

      if (mathSymbol === "+"){
            console.log(num1 + num2);
      }

      reader.close()

    });
```

Build Out the Calculator

Finish the calculator so that it handles subtraction, multiplication, division, and square root. You may need to search the internet to learn how to complete the proper mathematical operations in Javascript.

For each new operation, add a new if-statement following the same pattern as the addition if-statement above.

In order to practice your git skills, after you get each new math operation working, use ***git add*** and ***git commit*** to save your work. Go ahead commit your code now, since you added code that allows the user to add two numbers togeter.

```
$ git add calculator.js
$ git commit -m "Addition implementation working"
```

Note: Read the Docs!

The Node.js official documentation will be very helpful to you today, and moving forward in this course. You can always google and read other resources, but if you're ever looking for a good general reference and this site pops up in your search results, it's a good one to use.

Node.js Documentation

Do not move on to the next section until you have finished implementing subtraction, multiplication, division, and square root calculator functionality.

Backing Up Code to GitHub

Git is great and all, but your files still just exist on your computer which is nice… as long as your computer works. Enter GitHub!

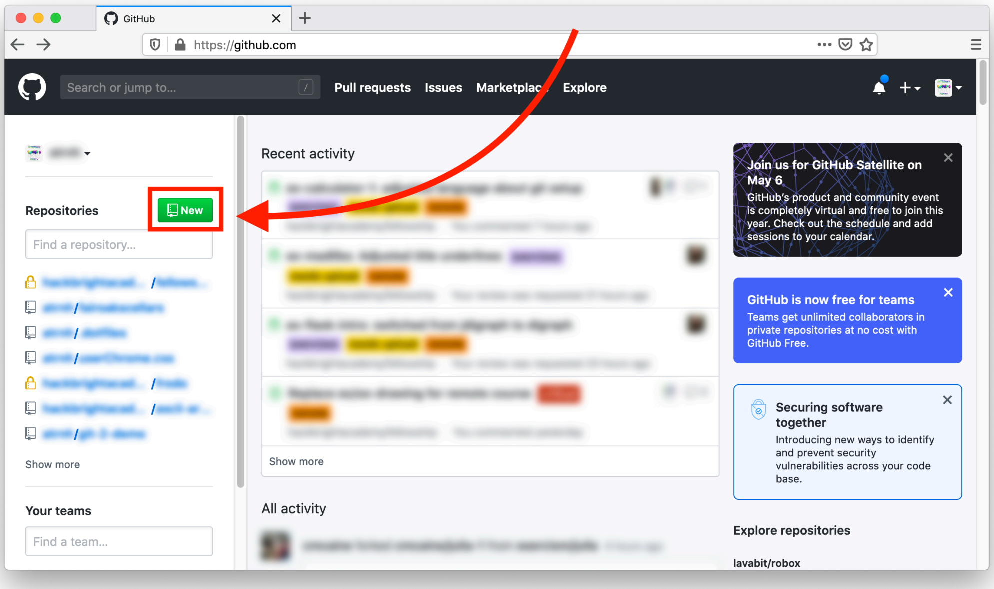
GitHub is a company that gives developers a place to upload their code, showcase projects in a portfolio, and enable collaboration between developers by taking advantage of Git to sync code between different machines. You don't need to have a GitHub account to use Git but a GitHub account *will* allow you to make your code and yourself more visible in the software engineering

community.

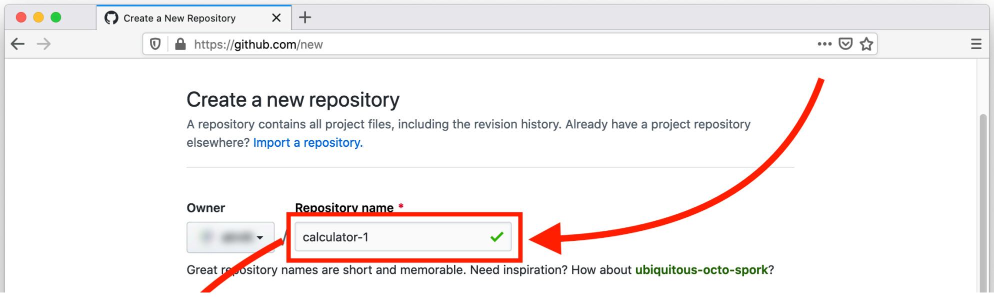
Warning: Do you have a GitHub account?

If you have not yet created a GitHub account, click here to create one now.

Every time you start a new software project, you should upload it to GitHub. Let's start by uploading your code from this exercise. To get started, log into GitHub and click the green ***New*** button to create a new GitHub repository (see the screenshot below). You'll want to create a new GitHub repository for every Git repository you want to upload.


GitHub
https://github.com
Search or jump to...
Pull requests
Issues
Marketplace
Explore
Repositories
New
Find a repository...
Show more
Your teams
Find a team...
Recent activity
Show more
All activity
Join us for GitHub Satellite on May 6
GitHub's product and community event is completely virtual and free to join this year. Check out the schedule and add sessions to your calendar.
GitHub is now free for teams
Teams get unlimited collaborators in private repositories at no cost with GitHub Free.
Securing software together
Introducing new ways to identify and prevent security vulnerabilities across your code base.
Explore repositories
lavabit/robox

This will take you to a form to create a GitHub repository:



Create a new repository

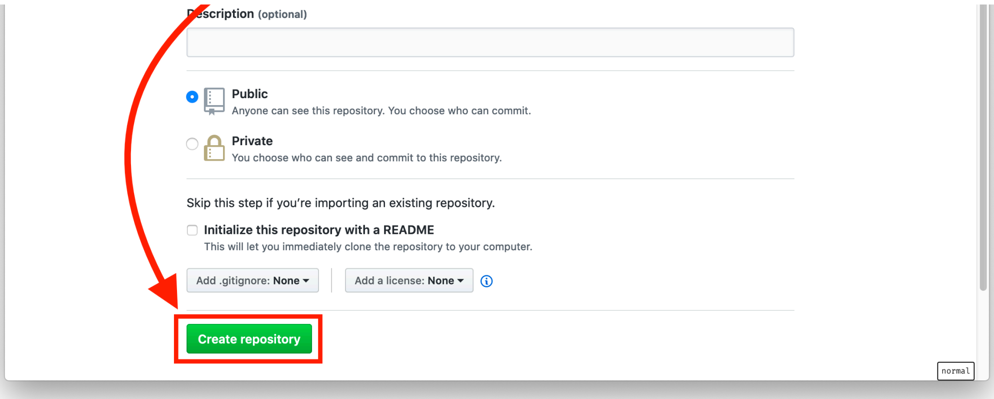
A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Owner

Repository name *

calculator-1

Great repository names are short and memorable. Need inspiration? How about ubiquitous-octo-spork?



Description (optional)

[✓ Public]
Anyone can see this repository. You choose who can commit.

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

Skip this step if you're importing an existing repository.

☐ Initialize this repository with a README
This will let you immediately clone the repository to your computer.

Here’s how to fill out the form:

- Under ***Repository Name***, enter `calculator` (or whatever you’d like call it, but note this affects the URL)
- The description is optional
- Leave it set to ***Public*** — we want your future employers to be able to check out your code
- Leave ***Initialize this repository with a README*** unchecked — you’ve already initialized the repository on your local machine with ***git init***

Click ***Create Repository*** to create a repository with the above settings. You’ve just created a ***remote repository*** (“remote repo” for short or just “remote”). It’s a remote repository (think like a *remote island*) because it exists far away from your computer. The repository on your computer is a ***local repository*** (or “local repo”).

Note: Industry terms

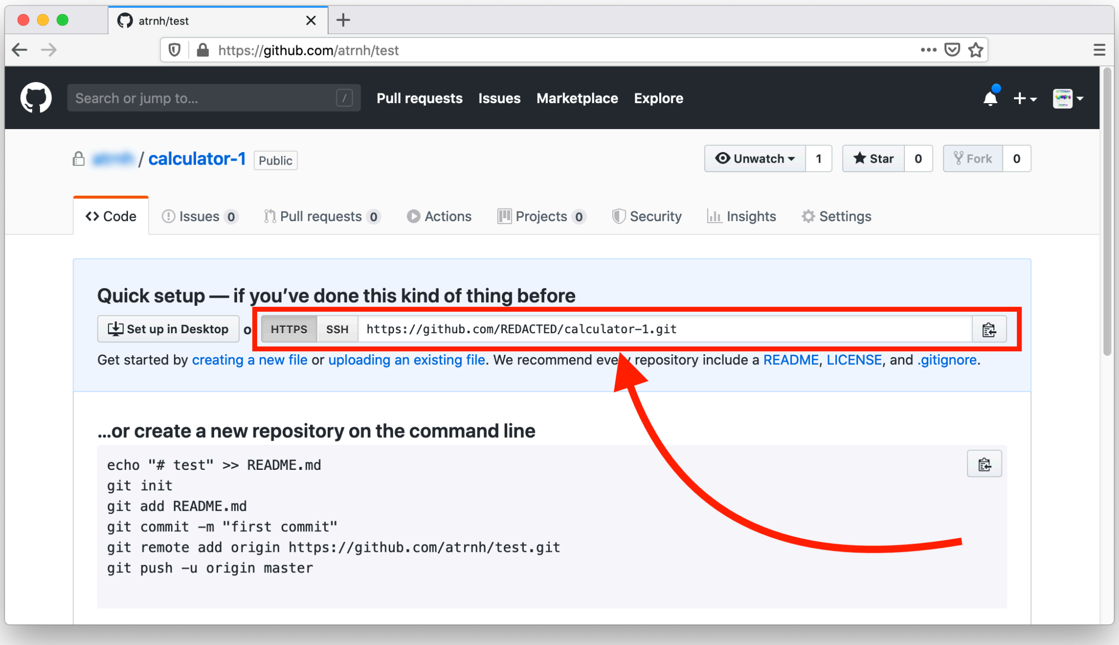
“Remote repository” is a mouthful, so most of the time people use the term “remote repo” or, even more informally, “remote” instead. For example:

- Did you check what’s in the remote?
  - Translation: Did you check to see if the code you’re looking for is in GitHub/the remote repository?
- I can’t fetch from the remote repo
  - Translation: I can’t get the latest code from the GitHub repository.

Similarly, “local repository” is shortened to “local repo” and sometimes it’s shortened to “local” but only in certain contexts. For example:

- The updates are only local right now
  - Translation: My latest code exists locally, on my machine (implying that I haven’t uploaded it to the remote repo yet)
- I have a local repo with the files you’re looking for
  - Translation: I have a copy of the repository on my computer that contains some files that probably went missing, somehow
  - In this case, you wouldn’t say, “I have a local with the files you’re looking for” — that just sounds weird!

Now you’ll need to connect your remote repository to your local repository. To do this, you’ll need the address of the remote repo. From your remote repository’s homepage, locate the ***Quick Setup*** box and copy the HTTPS URL of your remote repo:



Quick setup — if you’ve done this kind of thing before

```
https://github.com/REDACTED/calculator-1.git
```

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.

…or create a new repository on the command line

```
echo "# test" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/atrnh/test.git
git push -u origin master
```

With your remote’s address in hand, head over to your terminal — next, you’ll tell your local repo that it should track your GitHub repo with the Git command, ***git remote add***.

Warning: Make sure you’re in the right place

Make sure you’re in the directory of your Git repo (it should be ***~/src/calculator-1***) before you execute any of the Git commands below.

git remote add takes in two arguments:

- The name you want to use to refer to your remote
  - The name can be anything you want but `origin` is the name everyone uses by default
  - This allows you to connect your local repo with *multiple* remote repos
- The remote's address

Putting all of the above together, here's the command to connect your GitHub repo and local repo together:

```
$ git remote add origin REPLACE_THIS_WITH_YOUR_URL
```

When we add a remote, much like when we do any other configuration, it only exists for that repository. Outside of the folder our repository is in, none of these settings exist.

Now we have a remote — this is like a portal that goes to whatever URL we point at. So, now that we've established a portal link, what do we do with it? Same thing you'd probably do if you actually opened a portal to the server room at GitHub. Push things through it and see what happens.

```
$ git push REPLACE_WITH_REMOTE_NAME main
```

Replace ***[remote name]*** (no square brackets) with the name you used when you added the remote (it's probably `origin`). You might wonder about what `main` is — that's the name of the branch. For now, we'll just have one branch, so we'll always type `main` there. In the ***Further Reading*** section, there's some information about branching.

Once you've ***git push***-ed, Git will ask you for a username. This is your GitHub username. Then it will ask for your password. (When you type a password, you may be used to seeing asterisks (*****) appear; in this case, nothing will. Don't worry! It's still working. This is just another way of hiding your password.)

After this, head over to your GitHub profile. You should see that your new repo has been created, and that any files you've added have been pushed to the server. Pushing code to

GitHub is the simplest way to back up your work to the cloud, share code between teams, and make sure you remember what you did and when you did it (complete with notes to yourself!).

Congratulations on completing the Calculator!