Building a Great GitHub Profile

(https://github.com/learn-co-curriculum/git-github-github-profile) (https://github.com/learn-co-curriculum/git-github-github-profile/issues/new)

Learning Goals

- Create a professional-level GitHub profile
- Answer interview-style questions about your code

Introduction

After you graduate from Flatiron, especially if you're going to be looking for a job as a developer, one of the most important aspects of your professional identity is your GitHub profile.

The good news is that you've learned a lot of the things that make for a good GitHub profile in this section! Things like following good practices for making commits and creating commit messages, using branches, and using collaborative workflows will all show you have professional-caliber Git/GitHub skills.

In this lesson, you will learn some additional tips for making your GitHub profile shine.

Personal Info

At a minimum, your profile should include the following;

- Photo: how formal your photo needs to be will depend to some extent on the industry you want to get into — a formal photo will be more appropriate with a big financial services company than with than a tech startup. Regardless of the industry, however, make sure it's professional.
- Bio: here you can focus on your technical skills and knowledge, more so than you probably would on LinkedIn.
- Contact information: Email, LinkedIn, other social media, online portfolio, etc.

To really stand out, you can also create a readme for your profile. This will allow your profile to more fully represent who you are and what makes you unique.

Here are some resources you can consult to learn how to create a profile readme and get ideas:

- **Create Awesome Git readMe Profile (https://medium.com/swlh/create-awesome-git-readme-profile-84efa0bcda3b)**
- **GitHub Revamp Guide (https://benthedunn.medium.com/github-revamp-guide-8f48a890e61e)** (Flatiron grad)
- **Awesome GitHub Profile README (https://github.com/abhisheknaiidu/awesome-github-profile-readme)**

Contributions

The contributions graph is one of the first things recruiters look at. Having a graph with lots of green is an indicator that you are passionate about coding. But having a colorful graph isn't enough — the specifics of your contributions are just as important.

Here are some of the factors that recruiters look for in your contributions:

- Do you work on projects of your own, not just forked projects?
- Do you work collaboratively with groups?
- Do you make contributions to open source projects?
- Do your open source contributions include actual code, or just things like typo fixes or minor edits to documentation? (These types of contributions are fine, but you don't want your contributions to be **only** of that type.)

One thing to be aware of is that contributions you make do not always show up in your contributions graph or contribution activity. Check out **this GitHub documentation (https://docs.github.com/en/account-and-profile/setting-up-and-managing-your-github-profile/managing-contribution-graphs-on-your-profile/why-are-my-contributions-not-showing-up-on-my-profile)** that explains why that is and how to fix it.

Finally, you can make your GitHub activity more public by **linking your repos to your LinkedIn profile (https://towardsdatascience.com/how-to-feature-your-github-repositories-on-linkedin-56078e1ffddb)** .

Pinned Repositories

The pinned repos is where you can highlight the projects you've worked on. It's where you can show the skills you've developed and the range of technologies you can use. However, while showing you know a diversity of technologies is good, you don't want your pinned repos to be all over

the place. Make sure you focus on the particular technology you're interested in working with, or the particular type of job you're interested in (e.g., front end vs. back end).

Choosing repos to pin:

- Make sure the repos are high quality: they should not be broken or buggy, and the code should be clean and **DRY** (https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) (Don't Repeat Yourself). It is better to have a small number of high quality pinned repos than a lot of repos of lower quality.
- Make sure there's at least one that you've contributed to recently.
- Include projects you've worked on rather than labs or tutorials.

All pinned repos should have:

- A **solid ReadMe** (https://www.freecodecamp.org/news/how-to-write-a-good-readme-file/), including:
  - Demo video and deployment link,
  - Gifs (preferably) or screen shots,
  - Extras: wireframes and user stories (planning materials), a blog post that you wrote about your project.
- A description and a deployment link.
- Commit messages that are properly syntaxed and professional.
- File structure convention specific to the technology (e.g, a React app should include Components, Assets, and Styles folders).
- Clean Code:
  - The code should be specific to what you're trying to do and there shouldn't be extra code (e.g., from a scaffold) that isn't serving any purpose. You should be able to explain what your code is doing and why it's there.
  - Remove console.logs!
  - Code should be properly structured: proper indentation, etc.
  - Functions should follow the **Single Responsibility Principle** (https://en.wikipedia.org/wiki/Single-responsibility_principle); aim for functions that consist of 2-5 lines of code.
  - Variables should have names that describe the data they store, and functions should have names that describe what they are doing. Do not use comments in place of good naming conventions!
  - Code should be **DRY** (https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Use helper functions for tasks that are performed more than once. Remove code that is not being used.

- Code should be conscientious with data: you should not have unnecessary fetch calls, or pass or store data that isn't needed.
- There should be a good balance between making your code tight and keeping it readable.

Answering Questions about Your GitHub Profile

When the time comes for you to start interviewing, you should be prepared to answer questions about your GitHub profile. Here are some of the types of questions you can expect:

- Are any of your commits contributions to an open source project/group collaboration/your own repos?
- Did you research standard file organization for the technology?
- Would you be able to explain why you set it up the way you did?
- How much of this did you write yourself vs. generate with a shortcut command?
- Did you remove all of your console.log()s?
- Did you have someone else review it?
- Do any of the functions exceed 2-5 lines of code?
- Is the code easy to read for you?
- Can you get a sense of what it’s doing from the variable naming and structure?
- Is there any code that is not doing anything?
- Could you explain what every piece of code is doing and why it's necessary?
- Are there any repetitions in your code? Could it be refactored into a helper function?
- Is your code following the **single source of truth principle (https://en.wikipedia.org/wiki/Single_source_of_truth)** ?
- Are there any functions that have access to data that they don’t need?
- Are there any components with unnecessary state/props?

Tips for answering questions:

- Explain your code as you would to a 10 year old, or your non-technical coach!
- Acknowledge when you don’t know something and keep it moving forward: “I don’t know, but will use xyz resources to find out”; or, “I would strategize and prioritize to make xyz more efficient”.
- When answering questions, walk through your thought process: employers want to know how you think through problems.
- Practice answering these types of questions with a friend or coach!

Exercise

Find a fellow student, instructor, or coach and ask them to review one of your projects and ask you questions about it. Practice using the tips above as you answer.

Conclusion

A lot of creating a good GitHub profile is developing good Git and GitHub habits and good coding practices from the start. These habits and practices will make you a better programmer, and will show up in your code and your GitHub profile. Nobody can meet all of these expectations all of the time, but the sooner you start working on it, the better off you'll be!

Additional Resources

- **Flatiron School Webinar: What Makes a Strong GitHub? (https://www.youtube.com/watch?v=jUYQPl2RUpw)**

(https://www.youtube.com/watch?v=jUYQPl2RUpw)