Lesson Wednesday

Introduction to Programming (/introduction-to-programming) / Git, HTML and CSS (/introduction-to-programming/git-html-and-css) / Homework: Preparing for Your First Code Review

Text

What To Expect

The primary goal of every code review is learning and practice. While the format will prepare you for code reviews at your future workplace, at Epicodus, code reviews are an opportunity to solidify the core concepts you've learned in a course section. Because of this, the code review prompts are generally easier than the practice prompts that you encounter towards the end of a course section (which are designed to push your comfort level).

However, **we do not expect students to perfectly understand or apply the concepts from the course section.** We know the time it takes to understand a concept varies, and may require a conversation with your instructor. It can be unnecessarily demoralizing to feel like you are falling behind because you *think* you did not do well on a code review. In these cases, trust your instructor to let you know how you are doing. As instructors, we commonly see that students feel badly about their performance when there's no reason to. If you need reassurance about your work and progress, please reach out to your instructor.

We know that some students have testing-related anxieties, and other students are getting back into a structured educational setting for the first time in a while. Both of these (and other anxieties) can make it more challenging to do well on a code review. If you can relate, please keep in mind that there's always the opportunity to revise your work after you receive feedback and direction from your instructor. It's OK to not get things perfect the first time around. We also see that some students struggle on the first few code reviews, but do fine thereafter because they've gotten used to the assignment and assessment structure at Epicodus. Whatever your experience is, if you feel like you are struggling, check in with your instructor to get some reassurance.

Our biggest expectation for students is to use the time during a code review to develop their general research and creative troubleshooting skills. Developers hit walls all of the time, and you will, too. The best tool to move past a roadblock is to be persistent and continue troubleshooting! The more you spend time developing these skills, the stronger they will become. In the long run, this will help you become a self-sufficient developer who is able to take on more complex and ambiguous programming tasks.

We have some practical strategies for approaching code review to share with you in this lesson. Continue reading, and bring any questions or concerns to your instructor.

Strategy #1: Focus On the Minimum Viable Product

In web development, **MVP** stands for **minimum viable product**. As the name implies, this is the *minimum* or *most basic* version of an application. The MVP must solve the problem the application is meant to address — but only in a basic form, without added bells and whistles such as styling, more advanced user interface (UI) features and other stretch goals.

MVP Example

Let's consider some examples of MVPs. For instance, consider the Wright Brothers' famous first flight in 1903 — it lasted under a minute and the Flyer traveled 852 feet — and was badly damaged in the process. The Flyer was so unstable that only the Wright brothers could get it in the air at all.

But that was the MVP — they did achieve flight, even if only for a very brief period of time. At that point, the Wright brothers began thinking about improvements they could make to their design. How could it fly longer and higher? How could it be more stable? How could they improve the UI so others could fly their machines? These are all stretch goals beyond the MVP.

From the perspective of designing a web application that's an MVP, it's typical to go through the following process:

- Build an MVP that does the thing it's supposed to do — whether it's flying or teaching customers how to make pizza.
- Once the MVP is done, talk to potential users and customers. What can be done to make the product better?

MVP Benefits

By following this MVP process to create our product, we reap two distinct benefits:

- **We have a functioning product completed sooner**. While it may not have all the fancy features we envision, it's better to prioritize building a functioning prototype with fewer features than to add too many features at once, which can result in delays and missed deadlines. You can always add those extra features after your basic MVP is finished.

- **We can also gather user feedback before adding extra features**. After creating an MVP, users can beta-test it and provide feedback. This allows you to determine what additional features users actually want.

For code reviews at Epicodus, the first benefit is what matters most to us: getting the project done on time. It's easy to get side tracked in general, not to mention side tracked on the bells and whistles of a project. There's also room for your creative direction in code reviews, and we encourage students to have fun with the styling and logic of their websites. However, the downside of this is getting too focused on features that are not a part of the core required functionality, which often results in not completing the project by the end of the work session. Because of this, we strongly suggest that you focus on building your project's MVP first. After that, you can revisit your code and make it more complex, or make improvements to your UI.

Every code review prompt has a theme and specific required objectives, which outline the MVP of the application. To stay on track with the MVP, some students find it helpful to create a list of objectives from the CR prompt that they can check off throughout the day. Other students find it helpful to review the prompt requirements at the beginning and end of their work session to ensure they've met all of the required objectives.

Do what works best for you, but make sure to focus on the MVP first.

Strategy #2: Learn From the Bugs You Encounter

Running into bugs is a part of the development process. A quick search for web development jokes returns many memes and comics that lament the trials of facing bugs. Sometimes fixing a bug can take hours or days and there's a high likelihood of feeling frustrated. If the bug turns out to be due to a typo, you may even feel a bit silly after solving it. However, this is a normal part of programming, especially when you are just starting out!

Hands down, working on a bug makes for some of the best learning experiences. Sure, it will be frustrating, but debugging is a vital part of developing your troubleshooting and research skills.

What To Do When You Hit a Wall

When you are working on a bug for a while, or you are just not sure where to start troubleshooting, this is what we suggest:

1. **Take a break**. Go for a walk, eat a snack, and do whatever you can to get your mind off of the problem.
2. **Come back to the bug.** Returning to a problem after a break can help you see your code in a brand new way.
3. **Be persistent**. Try out multiple different troubleshooting techniques (see below), one at a time. Keep a record of what you do. Generally, trying out one troubleshooting measure at a time is best to avoid potentially making the problem worse, or solving the problem without knowing why.
4. **Get help**. If you've exhausted your troubleshooting ideas and cannot solve the bug, write a submission note for your instructor about the issue and all of the things you've done to try and solve it. If possible, move onto a different aspect of the project.

Different Troubleshooting Techniques

Review the context of your error. What did you last do? Undo your last change (with ctrl + z or cmd + z) — does this undo the bug? It can be helpful to trace your last steps. **Read and research the error message.** Make sure to pay attention to the date listed on any resource you find. Solutions can be outdated, or platform-specific (for example, specific to a browser or operating system). Error messages often have a line number potentially pointing to the problem in the code. **Research solutions online.** If you are hitting a wall, look to documentation online for different ways to achieve your goal. You might come across a different tool or method that helps you achieve your coding goal. Generally speaking, avoid using code that you find online that you do not understand. **Look at official documentation.** If you are executing a series of steps to do something (like creating a gh-pages for your website) and you are not arriving at the expected outcome, it can be helpful to review official documentation.

Strategy #3: Review Your Work

This is as straightforward as it sounds: at the end of your work session, review the prompt and your project to make sure you've completed all requirements. While you don't know what you don't know, reviewing your work against the project requirements can help you find things you've missed or forgotten.

In the end, it is your responsibility to complete the project requirements, and you can do that whenever you have the time. If you know something is missing by the time you need to submit your project, you can submit your project *and* continue working on it. You also don't have to wait for instructor feedback to continue to work on your project. Simply continue to commit and push your changes to your remote repo, and your teacher will see those changes.

Summary

In this lesson, we covered general expectations for code reviews and common challenges we see. We also reviewed three strategies that can help you be more successful in your code reviews:

- Focus on your project's MVP
- Learn from the bugs you encounter
- Review your work

Remember, the languages, tools, and approaches you'll learn here are much less important than the general skills of solving problems. If you hit a wall, this is normal. Take a break if you need to, and when you come back to the problem, be persistent and draw on all of your resources.

Previous (/introduction-to-programming/git-html-and-css/researching-online-and-helpful-resources)
Next (/introduction-to-programming/git-html-and-css/homework-git-expectations-for-independent-projects)
Lesson 49 of 64

Last updated February 28, 2023

disable dark mode

Epicodus (http://www.epicodus.com)