How to ask technical questions

It is a common misconception to keep your questions short and concise so that you do not waste anyone's time, however, there is usually a lot more to your question than you realize. When technical questions are missing key details, it is almost impossible to provide accurate answers. Therefore, we expect everyone to put forth effort into asking detailed questions that contain all of the relevant information.

Everyone in our Discord community is a volunteer, so you want to make it as easy as possible for others to help you. We expect everyone to learn how to ask detailed questions, which will equip you to ask better questions in, for example, our Discord, on Stack Overflow, and in your future workplace!

Before asking a question

Do your own research

Before asking a question, you should always do your own research. Use your favorite search engine, read forums like Stack Overflow, and use Discord's search feature in our server. If your search results are not helpful, adjust your search terms to find more beneficial results.

Help yourself

For many problems that you run into, you are capable of figuring out how to solve them. You are just missing the experience to know what to do or where to look. Read our practical tips on How to Help Yourself Before Asking Others. Over time, you will begin to incorporate these tips into your development workflow.

Pick the most relevant help channel

We have specific channels for every section in our curriculum, so please find the most relevant help channel to ask your question, like **#git-help**, **#html-css-0/#html-css-1**, **#js-help-0/#js-help-1**, **#ruby-help**, etc. When in doubt, you can ask curriculum-related questions in **#odin-main**, but it can get really busy in that channel so posts get buried quickly.

Prepare your question

Asking detailed technical questions will take time to prepare all of the relevant information. This investment of time will make it easier for others to help you and will also help you understand your problem even better.

1. Provide a link to the lesson/project in the curriculum

This is important because it provides the surrounding context to your question. In addition, knowing where you are in the curriculum will show your current knowledge and skill level.

2. Provide code, pseudo-code, or other relevant information

When asking a question it is essential to provide your code, error message, terminal command, server output, etc. If you have more than a few lines to share, you should use an external service:

- CodePen for basic HTML/CSS/Javascript
- Replit for Javascript/Ruby
- CodeSandbox for Webpack/React
- Pastebin for error messages or server output

If you have a question before you have written any code, you should **provide your pseudo code**. If you need help writing pseudo code, you should read through our Problem Solving lesson with your specific issue in mind.

Sometimes it might make sense to **share a link to your GitHub repository**. Be sure to make it as easy as possible for others to help you by providing detailed information (file names, function/method names, line numbers, etc).

When you need to share an image of your terminal window or computer screen, **share a screenshot** using a screenshot tool on your computer. If you are not familiar with this tool, you should research it using your favorite search engine. **Do not share a photo of your computer screen.** Pictures of computer screens are low quality images that are difficult for others to read.

Do not share files that require a user to download them. It is impossible to tell whether a file contains malware or has some other malicious intent. Therefore, you should never share a file that requires users to download and open it on their own device.

3. Explain the problem

Explain the problem that you are having and include any steps needed to reproduce the problem. In addition, you should also explain what part of the project you are trying to solve. This is important so others will know if the direction you are going will solve that part of the project, or if it would be better to steer you in a different direction. This is commonly referred to as the "XY Problem", which is a common pitfall for programmers as they learn how to ask detailed questions.

4. Describe what you are expecting

With as much detail as possible, describe what you are expecting to happen and why you are expecting this to happen.

5. Summarize what you have tried

Summarize your theory of what is happening, what you investigated, and the results you had after implementing any possible solutions. Explain any findings that you discovered while you were debugging, especially if you were able to pinpoint where the unusual behavior started happening.

After Posting Your Question

After you have prepared your question and posted it, there are a few more things that you need to keep in mind.

Help may look different than expected

Most of the time, your question will not just be provided with an answer. Instead, you will have a back-and-forth conversation leading you to your own answer, a lesson to review, a resource explaining something in more detail, etc. If this approach is different than you expect, you should read How to Help Others Solve Coding Problems.

Stay online and wait patiently

Stay online after asking a question. Since answers are not typically straightforward, you should be prepared to stick around and discuss it with those trying to help.

Wait patiently for a response. Even though our community is fairly active, not all questions will be answered right away. While you are waiting, make sure that you have provided a detailed question and continue troubleshooting. It is recommended to edit your post with the results of things you have tried. If your post gets buried, you may repost your question, otherwise, you may direct people to your question in a more active channel.

Be patient and willing to provide more details

If you ask a question without providing enough details, you will be asked to provide all the relevant information. We occasionally see people taking offense or being impatient when asked to provide more information. Be assured that these questions are part of the process to avoid the frustration of helping someone with an incomplete picture. Your question may seem obvious to you, but it is often not obvious to others.

Avoid asking follow up questions

Once your initial question is answered, resist the urge to immediately ask follow up questions. Instead, you should begin this process over again by doing your own research. We realize that you mean well, but it is somewhat inconsiderate to expect the same person to continue helping you without making an effort on your own to understand it first. This is a form of handholding that our community discourages.

We discourage low-effort questions and hand-holding

Our curriculum and Discord community highly value helping you stand on your own two feet without holding your hand. Adjusting to this approach can be difficult because it is different from the majority of your educational experiences. If you develop a pattern of asking low effort questions, one of our moderators will privately address this with you because these types of questions cause a drain on our community of volunteers.

Conclusion

One of the biggest benefits of taking the time to write a very detailed question is that you will occasionally solve your own problem. This is one of the best outcomes to have! You should celebrate this achievement and consider documenting the problem and solution in the README.md of the project that you are working on.

It is highly recommended to review these tips each time you ask a question, at least until it is second nature to always provide this level of detail. If you'd like to read more about providing surrounding context, you should read Stack Overflow's standards or one of our favorite articles, How to be Great at Asking Coding Questions, by Gordon Zhu.

Report an issue