Jumpstart TA Handbook

Thank you for volunteering to TA at Jumpstart! New students get to see how far they can go in a couple of short months and it is so inspiring. We know you will do a great job! Here are a few guidelines for being a successful and effective TA this weekend.

You are representing LEARN

1. Follow LEARN's code of conduct at all times.
2. Be mindful of your language.
3. Hold others accountable for their conduct.
4. Report any issues or concerns regarding fellow TA's or students to the instructor/s.

Remember, the people attending Jumpstart are *beginners!*

1. Put yourself in a beginner's mindset. Ask yourself what was confusing to you the first time you learned about it? How would you have liked it explained to you?
2. Keep it as simple as possible. All the problems can be solved with very basic code (i.e. if/else statements, lots of variables, etc.) You don't need arrays, loops, higher order functions, etc. Of course you *could* use these things but you don't *have* to, so let's not make things too complicated.

3. It can be tempting to want to show students your advanced, elegant ideas! However, this can be deconstructive and overwhelming for people who are new to coding.
4. Be clear in your language. Define terms as you use them.
5. Enforce good practices from the very beginning. Help students build good habits by helping with indentation, semantic naming, etc.
6. TA's should let the instructors instruct. Practice what the instructor preaches, stay on brand, don't disparage things like the course being online, any tech people are using, or any processes we set up. We should be a united front!

Tips for providing guidance

1. If someone is asking you for help, stay clear with your intent and keep a dialogue going that describes what you are looking for.
2. Try your best to stay in the navigator role when bug fixing. If you can't solve the problem from this perspective, ask someone else to come in as well and help. Avoid driving for them. It doesn't help them learn, and doesn't help you flex your mentor muscle!
3. Don't settle for work-arounds or code patches.
4. Do your best to keep their existing code without scrapping it. Sometimes when you walk into spaghetti code it can be hard, but avoid saying, "start over" or "delete everything." Keep refactoring until you get to working code while console.logging along the way. This will demonstrate the process of solving a problem, rather than delivering them magically working code.
5. When assisting students in their breakout room, we don't want to just *give* the answer to the student. We want to walk through the issue together, fully understand what it is they are trying to achieve with active listening, then work on their problem with them by providing hints in the right direction.
6. If the student is SUPER stuck, it's been a bit of time, and they're still frustrated, go ahead and get them out of frustration by giving more direct guidance.

7. When assisting a student in their breakout room, reinforce the idea that they should try their best for about 10-15 minutes on their own, by brute forcing the problem, googling then eventually calling for help.
8. It’s OK to say, “You know what, I’m not sure! Let me find out for you and get back to you in a minute with that.”
9. Try and get to know people's backgrounds and what they are interested in and why they want to learn how to code.

Managing your time

1. Pop in and out of rooms. If you sense frustration, let someone else know so we can help intervene.
2. Actively check in on students. Don’t spend too much time just ‘sitting around’ in either the main room or a single breakout room.
3. After helping out someone in a breakout room, make sure they are set off in the right direction and then let them know if they need anything to just pop into the main room and ask. After that, leave the breakout room and rejoin the main room.