Code Fellows

Teaching Assistant Handbook

Version 5

A guide to the inner workings of being a Code Fellows TA

Chapter 1

Welcome to Code Fellows!

Greetings and Congratulations!

Welcome to our team of world-class teaching assistants, we are glad to have you on board! This handbook is meant to help orient and teach you how and why we do things the way that we do. This handbook contains best practices and linked resources, it should be read before, during, and after you assist for a course.

The View

When using this document, we recommend taking advantage of the Document Outline feature **(View > Show document outline)**. This feature will help you navigate the handbook as you're scrolling its pages.

Also, view the pages in Print layout (**View > Print layout**) so that all page numbers, headers, and footers are visible.

The TA Guiding Principle

A guiding principle is a statement that helps to guide our decisions and the way in which we approach training. As a TA, your guiding principle is to lead with humility and empathy to create safe and inclusive environments where people learn and lives are transformed.

The Code Fellows Way

Based on educational theory (***andragogy**: the method and practice of teaching adult learners*), industry standards, professional best practices, and LOTS of iteration, we have identified effective instructional methods.

Code Fellows is centered upon **empathetic teaching**. We believe empathy is the most important skill you can practice and will lead to greater success personally, professionally, and as a bonus, make you happier the more you practice.

You were hired as a TA because we detected some degree of empathy in you already. In order to help you grow this skill, please read the following articles:

- Your Most Important Skill: Empathy by esteemed developer, Chad Fowler.
- Learning Styles and Experiential Learning Cycle by David Kolb

We are aligned with student goals: Our students have big plans and goals, typically involving career and life changes. The position of TA is a leadership role that we actively encourage you to take ownership and command of within our program, which allows you to take accountability for the experience of our students.

We utilize a non-traditional instructional style: This style is more like mentoring, or coaching, or a personal trainer. We expect students to learn by doing, and to teach each other. Class time is discussion-oriented, to unblock student progress and thinking, to address common hazards, and lightly introduce new topics. Stacked learning is our overriding methodology, familiarize yourself with this concept here.

We are ruthlessly focused on practical skill: Code Fellows is in constant dialogue with industry partners to stay current with the abilities and aptitudes that are needed **now**, and to read the winds on technologies, concepts, and frameworks that are gaining traction. As a result, our curriculum is in a state of constant evolution. Pair-programming and team projects in our courses are designed to emulate real-world work environments. We emphasize professionalism, accountability, and communication skills alongside our technical curriculum, striving to produce graduates who can immediately provide value for employers and become key team players amongst their colleagues.

Why We Do This

We have a singular goal: Great people into great jobs! People come to us, looking for fulfillment of their hopes and dreams and they are making a massive investment of both time and energy to be here. It is deeply rewarding to help people learn new skills that serve to improve their lives.

What We Do

Students will expect you to know about the inner workings of Code Fellows. The FAQ page here is a great place to start.

Don’t worry if you cannot answer a question. Take the lead by helping the student find the answer, and learn it yourself, so you can answer next time. If a question is related to course discounts and/or is regarding specific aspects of the admissions process then please have students email admissions@codefellows.com to get the correct answer(s) from the Admissions department. If a question is related to billing, invoices and/or other financially related issues then please have students email finance@codefellows.com to get the correct answer(s) from the Finance department. Always be willing and ready to learn from your interactions with students! The best educators are constantly learning new things.

TAs should understand our admissions processes: students apply via the website, do an initial phone interview, prepare for their entrance test, take the test online via Canvas LMS), and are enrolled if they achieve the minimum passing score of 80%. Students with more advanced skills are able to skip courses by testing into the intermediate or advanced courses and successfully completing an in-person interview.

Familiarize yourself with our Course Offerings and Course Catalog, as well as the policies contained therein, found on our website:

Course Offerings

Course Catalog

TAs should know exactly what our Code of Conduct says and be able to answer student questions about what we consider harassment and how to treat others, as well as plagiarism, proper use of citations, open source practices, etc.

Chapter 2

Getting Started: Onboarding and Initial Training

Introduction to Our Tools

The team communicates in the following ways: via email, Slack and Google Docs. In the Code Fellows Slack workspace, each class has a private Slack channel for the entire class (by invitation from Code Fellows Developers admin account). Each class also has a private Slack channel for the instructor and TAs (again, by invitation). TAs are welcome to join and participate in any other public channels in the Code Fellows Slack workspace. In addition to Slack's web functionality, it also has a mobile app, and Mac desktop app. **Please ensure that**

your Slack profile is up to date with a professional looking picture and includes some bio info to help students get to know you.

Google Docs, Google Drive, or other tools may be used within a course, depending upon the instructor.

The team uses GitHub and Canvas for managing courses and students. The Principal Teaching Assistant will add you to the TA Team in the GitHub Code Fellows Organization. Classes will have repos on the master Code Fellows account that accumulate the code written in class, plus contain other code samples supplied by the instructor. TAs do not have admin permissions for this repo, but can contribute via Issues and Pull Requests. Students generally will submit assignments by links to their GitHub repos of coursework. GitHub Gists are also used to share code snippets at times.

Our learning management system, Canvas, can be overwhelming for first-time users thrown at it with no guidance. Moreover, we use it in a very specific way, to complement the assignment review we do in GitHub. Many specific aspects of Canvas require their own introduction (specifically important: using the Gradebook and SpeedGrader). You can learn a lot by just clicking around and exploring Canvas, and it is encouraged that you set aside some time to do so. Investigate each of the following components:

- Syllabus
- Roster
- Assignments
- Quizzes
- Gradebook
- Discussion forums
- Supplemental files

You will be invited to your course Canvas page by a staff admin once it has been set up; generally, the Principal Teaching Assistant is your first point of contact for this. To get set up properly, start by setting your timezone (the default is Mountain time, not Pacific!). There are a variety of notification settings that can also be applied, it is important to configure settings so that they are actually useful to you. Don't forget to Include a picture of your lovely face and a little bio info to help students get to know you.

Training Session

Prior to beginning work as a TA at Code Fellows, all new TAs must complete their TA Training course on Canvas.

Training includes:

- A review of this handbook followed by a short quiz.
- A review of the Grading and Expectations document followed by a short quiz.
- A review of the Code of Conduct document followed by a short quiz.
- Links to helpful Canvas tutorials on:
    - Using the Speedgrader.
    - Taking attendance.
    - Understanding the grade book.
- A 15 minute video where two of our instructors talk you through grading a students assignment submission.
- A short TEDx talk explaining the importance of growth mindset.

Shadowing

Though not always possible, it is encouraged that new TAs spend a portion of a lab session shadowing an experienced TA to get a feel for the role and have an opportunity to observe techniques used by experienced TAs. Student interaction operates on many levels, and each moment can be focused on a variety of topics: specific course content, the details of an assignment, the flow of the course, an issue with workflow (such as a Git or IDE problem), the mindset of a student, just shooting the breeze, and so on. By shadowing, a new TA gets to observe these interactions and consider how they would have engaged with the student. After shadowing, the TAs can reflect on the session and discuss the student interactions that took place.

Administrative Policies

Contracts and compensation: The primary point of contact for TA contracts is the Principal Teaching Assistant. Contracts typically run for the duration of the class days of the course. TAs get paid at the end of each month they work in.

What if a TA gets a job while a course is in session? It happens, and since most of our TAs are also alums, this actually makes us extra pleased for you! We ask that you please give the instructor, the Principal Teaching Assistant, and Campus Director as much notice as possible so that we can arrange for a replacement. Your contract will be prorated to account for the days you will be at your new job.

How to handle student concerns and grievances: If it is an issue of an academic nature, strive to confer with the instructor first. In the event it is an academic concern that would not be appropriate or helpful to report to an instructor, please report to the VP of Education. For all other issues, report to one of the following, depending upon the nature of the matter: the Campus Director, Principal Teaching Assistant, Director of Admissions, or direct the student to do so directly.

Chapter 3

Ideals in Action: Definitions of Success as a TA

As a Teacher

If, as a TA, you notice a student struggling due to pre-work not being complete or if the student did not adequately learn from it, notify the instructor either in person or via Slack immediately. The sooner these issues are addressed, the better.

When working with students on code, DO NOT simply take over their laptop and start typing. It is extremely important that each student maintains agency over their tools and their work. Think of it as a pair-programming exercise with the student as the driver, and the TA the navigator. Yes, it will take longer, but it also builds a lot of important skills in listening, thinking, typing, and coding.

Understanding the components of course structure

Assignments (reading and coding assignments):

- Assignments are structured by Day or Class Number.
- TAs should help the instructor ensure that dates are accurate.
- TAs have the ability to publish assignments, but this should be left to the instructor unless otherwise assigned.
- TAs should help the instructor keep to an appropriate schedule for publishing reading and coding assignments (often, reading assignments are published a week in advance, whereas coding assignments tend to trickle out more slowly).

- TAs should ensure there is a solid mental road map for the course; anything that is unclear to you will probably be unclear to the students, too, so communicate with the instructor about this.

Quizzes:

- TAs may be asked by the instructor to write quizzes.
- Quizzes are used to reinforce concepts from lecture.
- Set quizzes to autograde as much as possible (multiple choice, True/False, etc.).
- Allow the students unlimited retakes.
- Use their highest quiz score as their grade for that assignment.

Attendance:

- Attendance will be recorded daily in Canvas.
- Lead TAs have primary responsibility for recording attendance. In the event the Lead TA is absent for any reason, it becomes the responsibility of regular TAs.
- Students must be present for 90% of class sessions to pass a course.
- For daytime courses if attendance has not been taken by 2:30pm then a reminder will be sent to the relevant team Slack channel.

How to be part of a successful Project Week: Students will work in teams leading up to a pitch day, in which students pitch potential projects to contribute to for the remainder of the course. Once a set of projects is chosen, they should be scoped for a three to four person team over a week. TAs are encouraged to be part of this process. It is vital to work with students on the projects so that they can show a minimal set of functionality and demonstrate that they understand the concepts taught in the course. Students will tend toward ambition, it is on us to create realistic expectations. Minimal Viable Product (MVP) is the initial goal.

As a Team Member

Always reflect and then communicate your thoughts and observations to the instructor:

- What went well today?
- What can be improved for tomorrow?
- What are you doing to enhance your performance as a TA?

Regular teaching assistant meetings will be led by the instructor, where you'll report on how the week went from each person's perspective, discuss any students needing extra attention, and whether grading is up to date. If you haven't had at least one team meeting in any given week then politely bring this up with your instructor.

As a Human

Empathy!

Be yourself. There is a reason you have been recognized as someone who should be "on the bus". Students want to hear from you, your experience, and your opinions. Let your personality shine.

Don't pretend to the students that you know more than you do but also be confident so they have an anchor to depend on when the coding gets tough. Show them how you find answers to tough questions, so they can learn how to learn. Put the needs of your students above your own and remember that you are to lead them by serving them and hopefully provide a model for them to follow in the future. You are here to help them achieve their goals.

Chapter 4

Operations: Responsibilities & Expectations

Overview of Daily Activities

The listing of TA responsibilities in this section of the handbook will consider an example day in a variety of divisions. Depending upon the course and the needs of the instructor, an individual will likely only have duties in some subset of these timeframes: class time (before, during, or after), lab hours, and all other times.

Daily expectations of teaching assistants, in general:

- Be present and be engaged to help students as needed.
- Be available and approachable at all times during labs. Even if you are with a student and therefore unavailable you can still project a feeling of approachability.
- Grade and comment upon assignments within 24 hours of initial submission. Final Project Week assignments are graded by the Lead and/or Assistant Instructor (if the course has been assigned an Assistant Instructor).
- Understand the instruction plan of the day, and how it fits into the plan for the week. TAs should always know what is going on.

BE ON TIME. If you have an issue that is going to cause you to be late or miss a shift, contact your instructor and hiring manager as soon as possible, via Slack or other agreed method.

TAs will occasionally be assigned other duties while on-site and in a shift at Code Fellows, such as cleaning whiteboards, assisting staff in facilities support (rearranging a room, or getting it straightened up prior to an event), or other similar tasks. This is coordinated with the Principal Teaching Assistant. Always make certain that the instructor for your course is aware of these activities.

Preparation helps TAs to be more effective, therefore it is strongly recommended that the TAs work through the class assignments and code their own versions, in advance, unless provided with completed examples. This provides the TA a ready-made and familiar body of code to show students as an example and gives them specific insight into what the students are seeking to accomplish with each assignment. By being able to show students multiple working versions of code, we directly illustrate to them that there are multiple approaches to solutions.

On a weekly basis, your performance will be reviewed by the instructional staff. Here are some examples of things that will result in a better review, and things that won't:

You'll get a better review if you exhibit the following competencies:

1. **Professionalism:** examples include proactively checking in with students and exhibiting positivity.
2. **Craft:** examples include being able to effectively and accurately assist students on your own, giving good mentorship, having good time management skills, giving good feedback, and knowing when to ask for help.
3. **Growth Mindset:** examples include showing empathy and being able to give and take good candid feedback.

The following will result in the opposite:

1. Showing a lack of professionalism by: being late to class, being late on grading, not being engaged with students, not communicating well or early enough with the instructor, too many absences, or not being prepared for the day's course agenda.
2. Showing a lack of attention to quality by: grading without giving feedback, grading inconsistently,

3. If you’re a Lead TA, showing a lack of leadership.

On a monthly basis you’ll be sent a job placement survey, just like every other Code Fellows graduate. It’s very important that you submit this report accurately every month. If you are a TA or know you’ll be a TA in the upcoming month, choose the option “Employed at Code Fellows In-Field” for your placement status. In subsequent months, choose the selection that most accurately reflects your placement status (i.e. if you’re TA-ing again, choose the same option, if you stop TA-ing and are seeking a job, choose that option, etc).

Grading

Our goal is to prepare our students for job readiness. For us that means graduates of the 401 class—not just finishers. Keeping our placement rate high in the market requires that being a “Code Fellows Grad” equals a high standard to hiring companies.

We want them to test into the best class for them. They don’t have to run through every one of our programs and our goal isn’t to push a specific percent through to the next program. Over the next year we’ll see students at the end of each class exit into industry jobs. We will be able to provide some additional guidance to the best role for them based on the class (internship, Junior Web Dev, etc). This means that we’re different than most academic institutions that provide a certificate or let people “graduate” and pass them along.

Each course is designed to prepare students for the next course. However, it doesn’t mean that all students will be ready. Often, 20% of the students won’t be ready to pass without further prep because they are coming in with less knowledge and experience than other students. We want to help both groups, and to do so we need to set realistic expectations for them.

When we hear from hiring companies, they don’t just want people that are trained in tech. They also want candidates that are a good culture fit with their businesses. A well-rounded graduate can show:

- Meaningful life experience

- Diverse thinking
- People skills
- Leadership
- Problem solving capabilities

Our partners ask: “Who are the top 5?” They want to know from both a technical and soft skills perspective. Don’t underestimate your ability to help students be job ready on that basis as well.

Ensure that students have a good understanding of how their overall grade will be determined, according to the rubric.

Grading is a hugely important part of being a TA at Code Fellows. It is our primary means of providing feedback to the students, and these interactions are essential. Students want to know, and ***need*** to know how they are doing in our classes. If there are problems, we want to be sure that they are identified and addressed rapidly.

Depending upon the course and the preferences of the instructor, the grading work may be divided among TAs in a variety of ways. In most cases a course is divided into sections in Canvas, with each TA having primary responsibility for a section each day with regards to grading. TAs will often divide responsibilities for monitoring the gradebook for assignment submissions over a weekend.

All assignments are to be graded and commented upon within 24 hours of the due date, or 24 hours from the time of submission if an assignment is turned in late. Be aware that this 24 hour turnaround does not apply to resubmissions. This level of feedback is exactly why students come to take our classes.

Every single assignment should receive an individualized response, whether a sentence or two, a paragraph, links to helpful resources, or the like. Strive to give students encouragement and to acknowledge their efforts as frequently as possible. We want our students to know that we’re here to help them, and that we appreciate their dedication to their work.

To grade reading assignments first read the student responses thoroughly and carefully. If a student asks a question, BE CERTAIN to answer it in your response. If a student makes an observation that merits a reaction, then react to it. If a student seems to have misunderstood something, provide clarity. Encourage students to be thoughtful, thorough, and reflective in their responses. While avoiding being hypercritical and overly pedantic, help students to use the right terminology and organization techniques to express themselves clearly. This is part of guiding them toward well-rounded professional competency.

Code assignments are typically submitted as merged GitHub pull requests. Depending upon the student and the assignment, your response might require some combination of commentary, inclusion of Gist for a suggested code snippet, or cloning of the repo so as to provide detailed commentary and suggested techniques for refactoring.

Follow these guidelines:

- Review assignment requirements.
- Unless the assignment is small, it is best to clone the assignment and run it to see if it works as expected. If it does not, comment to that fact and require resubmission for grading. Unfortunately, students will submit faulty code in order to meet the deadline. This allows you to quickly filter out such submissions.
- Review the assignment in more detail. Make sure each requirement for the assignment is met and, in the case of pass or fail only grade, return assignments that do not meet the requirements to the student for correction and re-submission.
- Review the code for common errors and point them out in a pull request or via comments (in Canvas, we do not recommend making the comments on the student GitHub repo).
- Try to find something encouraging to comment upon to temper the corrections made to assignments. Challenge strong students and encourage the weaker ones.
- If an assignment needs to be resubmitted and has not been for a couple of days after the due date and initial review, mark it as incomplete.
- If a student is consistently making the same mistakes, provide them with a Gist, including a copy of their code with your comments and a copy of their code that you've

refactored the way you would have written it. This takes much more time to produce and deliver than common pull request comments.

- When plagiarism is suspected or identified, seek to identify the suspected source or otherwise be prepared to explain your suspicion, and then consult with the instructor. Plagiarism is a Code of Conduct violation and can result in release of the student.

General Guidelines on Grading

- Pass: 90%+
    - Quality of work.
    - Assignments turned in on time.
    - Contribution to Team.
        - Are they helping other people in the cohort?
    - Leadership/attitude.
        - Are they setting a good example someone would want to hire? On time, positive contribution.
- Fail: <89%
    - Quality of work is just OK, or lacking in completeness or presentability.
    - Assignments are spotty or late.
    - Contribution to Team.
        - Doing only their own work, not participating or helping.
    - Leadership/attitude.
        - Are they setting a good example someone would want to hire? On time, positive contribution.

Complete Submissions

When you are grading, you should only grade submissions that are complete. For a submission to be deemed complete:

- the student must answer the required submission questions, which might include:
    - How long did you spend on this assignment?
    - Describe your process of completing this assignment?
    - What was the hardest/easiest part of this assignment?

- the code must meet professional criteria as demonstrated in the class, so it is expected that the code will:
    - Build.
    - Pass a linter.
    - Pass a style checker.
    - Pass all tests as appropriate to the level of course.

If a submission is incomplete (in any way), 0 to 5 points can be assigned after a cursory look over the code to determine how much effort was applied.

Assignment Rubric

Points are granted to each assignment based on how the submission meets the following criteria:

	Met all assignment requirements	**Idiomatic style used**	**Proper git workflow utilized**	**Other adjustments**	**Total possible**
Lab assignment	Up to 6 points	Up to 3 points	Up to 1 point	-2 points if late, -2 to +2 points for effort	**10 points**

Late Work and Assignment Resubmission Policy

- At the time of the submission deadline:
    - Assignments that are not turned in or are substantially incomplete will be graded as **0 points**.
- After the submission deadline:
    - Any assignment previously earning at least 2 points can be resubmitted for regrading with **no penalty**.
    - Any assignment initially awarded 1 point or less can be resubmitted for regrading and will incur a **2-point penalty**.

    - For students transitioning to an immediately-following class, work must be completed by 5:00pm Tuesday after the end of class. Contract and deposit must be completed the following day.
    - Resubmissions are allowed at any time prior to the start of the next Project Week.
    - For students who need more time after the course, they will email a plan to the instructor with what's needed to be done and target submission dates. Instructor can approve the plan if it looks reasonable, and let admissions know.
        - For 201 and 301, we should communicate this as being a strictly "as-needed" policy.
        - Students must turn in any incomplete work prior to being accepted to the next Code Fellows course in sequence.
- For handling student work that's late due to illness or pre-arranged absences, please see the State Catalog, Section 8: Makeup Work.

Fudge points allowed for effort

- If someone spends 15 minutes, and hits just the minimum, you can optionally take off up to 2 points. Include a comment encouraging them to push their learning further by digging deeper into the task at hand.
- If someone spends 8 hours, and makes clear progress but does not meet all the criteria, you can add up to 2 points.
- If someone puts in meaningful effort (at least 3-5 hours) and hits stretch goals, or elaborates on the solution independently, up to 2 points can be added as optional credit.
- Sometimes, experimentation and attempting different approaches is more important than getting the right answer.

A great submission follows good idiomatic code style and best practices

- JavaScript code looks like quality JavaScript code. C# code should look like quality C# code.
- Proper use of whitespace:
    - Indentation.

    - Appropriate new lines between methods and logical sections of code.
- Well-factored functions.
    - Attached to objects, or scoped properly.
    - Meaningfully named.
- Meaningful variables names.
- Meaningful/standard folder names and folder structure (e.g., scripts, styles, partials, …).
- Meaningful/standard file names (e.g. index.html, server.js, app.js, ...).
- Consistent and appropriate case usage (camelCase in JS, kabob-case in html, etc).
- Appropriate level of comments, especially comments that explain WHY the line or section of code exists.

A great submission follows Git best practices

- Meaningful commit messages, that explain WHY—not what—the changes are.
- Proper use of branches.
- No commented-out code included.
- Reasonable number of commits.
- Proper files included/ignored (e.g.: no `.DS_Store` files or `node_modules` flotsam).

Code should be original and attributed

The above assumes that work falls within the guidelines of the Code of Conduct, or consequences there should be followed.

Step-by-Step Grading Process

1. Before you start grading:
    a. Create a directory for the assignment you are about to grade.
    b. In that directory, create directories for each of the students you are responsible for grading.
2. Grading:
    a. Did they answer all the questions in their comments on Canvas?
        i. If not then do not grade and instead request that they answer the questions on Canvas.
        ii. If yes then continue with the grading process.

b. Clone the student’s Github repository.
c. Does their submission pass the linter?
    i. If not and it is clear they made little effort:
        1. Award 0 points.
        2. Direct them to resubmit.
    ii. If not but it is clear they made an effort:
        1. Award up to 5 points.
        2. Direct them to resubmit.
    iii. If yes then continue with the grading process.
d. Check for meaningful variable names:
    i. Do they accurately and succinctly describe the data they are storing?
        1. If not:
            a. Leave a comment on their pull request commit message on Github.
            b. Deduct 1 point.
        2. If yes then continue with the grading process.
e. Check for meaningful function names:
    i. Do they follow industry best practice (camel case, describe the action they perform etc.):
        1. If not:
            a. Leave a comment on their pull request commit message on Github.
            b. Deduct 1 point.
        2. If yes then continue with the grading process.
f. Check for well factored functions:
    i. Is the code well factored commensurate with what they have been taught to this point?
        1. If not:
            a. Leave a comment on their pull request commit message on Github.
            b. Deduct 1 point.
        2. If yes then continue with the grading process.
g. Check for meaningful project structure:

        i. Are the folder and file names meaningful and follow best industry practice?
            1. If not:
                a. Leave a comment on their pull request commit message on Github.
                b. Deduct 1 point.
            2. If yes then continue with the grading process.
    h. Check for appropriate level of comments:
        i. Could their comments be improved significantly in terms of being meaningful, too excessive, too sparse etc.?
            1. If yes:
                a. Leave a comment on their pull request commit message on Github.
                b. Deduct 1 point.
            2. If no then continue with the grading process.
    i. Check for code that has been commented out:
        i. Does commented out code exist?
            1. If yes:
                a. Leave a comment on their pull request commit message on Github.
                b. Deduct 1 point.
            2. If not then continue with the grading process.
    j. Check for proper git workflow:
        i. Do they have meaningful commit messages, proper use of branches, reasonable number of commits, a gitignore etc.?
            1. If not:
                a. Leave a comment on their pull request commit message on Github.
                b. Deduct 1 point.
            2. If yes then continue with the grading process.
    k. Check for late assignment submission:
        i. Was the assignment submitted past the deadline set?
            1. If yes:

            a. Deduct 2 points.
        2. If not then continue with the grading process.

l. Check for bonus points:
    i. Did they achieve any/all of the stretch goals specified for the assignment?
        1. Award additional points using best judgement.

Prior to the Start of Class Each Day

Circulate among students if you are there early, to see if anyone needs assistance.

If you are the Lead TA then verify that the instructor has everything needed to start class. If you are a supporting TA but still want to volunteer your assistance during lectures as well as your contracted lab assistance then check with your Lead TA for any tasks they may have for you.

During Class Each Day

Be aware and engaged with the class. Keep up with what the instructor is covering.

If a student needs assistance, be prepared to go help the student. Be mindful that your assistance does not cause the student to become disengaged from the class. If you are the Lead TA then make sure to take daily attendance on Canvas.

Otherwise, spend your time staying caught up on grading although most of the grading should have been done during the block of time during lectures that was agreed upon with your instructor.

After Class Each Day

Wait in the classroom for 5-10 minutes and be available for student questions and engagement. Once students have cleared out, break down the setup for screen recording if necessary. Help the instructor to erase the whiteboards. Make sure that the data projector has been turned off. Reset any furniture that has been moved back into its normal position. . If your instructor requests a video recording to be captured, contact your manager for assistance.

During Lab Hours

Circulate among the students regularly. Make sure to check in with every student in the class AT LEAST ONCE during each lab session. Take care to not get bogged down working with an individual student. It is your responsibility to help as many students as possible. Instructors might call a TA meeting during lab time. Otherwise, spend your time staying caught up on grading. Be available and approachable at all times during labs. Even if you are with a student and therefore technically unavailable you can still project a feeling of approachability so that the other student knows that you will certainly accommodate them once you become available again.

DO NOT wear headphones or earbuds during lab time. The only exception to this is if the Lead TA needs to listen to sound when making the YouTube lecture videos. Even in this rare case there is only need for one earphone/earbud to be used therefore indicating to students that you are still available to help them with lab related questions.

Availability at Times Not in Class or Labs (Including Weekends)

Keep an eye on the class Slack channel outside of regular meeting times, so as to address any student questions that arise. Encourage the channel to assist their colleagues on Slack when they post a question. This often means that as a TA you do not need to get involved and helps form relationships between the students that will serve them well for their time at Code Fellows. If the rest of the students are unable to help the student posting the question then offer them assistance but only after clarifying with the channel whether anyone else also requires help with the same issue. Spending 20 to 30 mins in a direct message chat with one student to then receive a message from another student right after with the same issue is not a good use of your time. If a number of students are stuck with the same thing then help them all in the main channel or create a group chat with all of the students who are struggling with the same topic.

You are welcome to offer weekend review sessions or office hours if you would like, but this is not required.

Depending upon the class, the instructor may ask the TAs to set up a rotating schedule to check in on Slack and to monitor the gradebook for assignments that need review.

Chapter 5

Conclusion

How Can WE Help YOU?

As a TA, you are a vital and valued member of the Code Fellows team. We want to do whatever we can to make the experience as rewarding, enjoyable, and enriching as possible. Be sure to read through the supplemental links at the end of this handbook. The more you learn about our educational processes and their underlying theories and practices, the better learner you will be. Good developers are always learning. Our instructors and Principal Teaching Assistant will be available to you during many of the off-hours in the event that you have any questions or observations.

Your Input is Welcome!

While assembling this handbook, we have tried to think of everything that you need to know. However, we know we have probably missed a thing or two. Plus, things change, and it is easy to overlook details from time to time. If you see anything in this handbook that could be explained more clearly, or have an idea for something to improve or add, let us know! Your primary point of contact for feedback is the Principal Teaching Assistant.

In Conclusion

A reminder of the TA Guiding Principle, “We lead with humility and empathy to create safe and inclusive environments where people learn and lives are transformed.”

Being a TA is a leadership position. Students look at you and understand that where they are now, you used to be and where you are now, is where they are striving to get to. Having that real life example in front of them, showing that it CAN be done is huge and with that also comes a hefty weight of responsibility to model for students what it looks like to be a Code Fellows Alumni.

Be sure to list your TA experience on your resume and on LinkedIn.

Have fun. Be professional. Be kind.

Appendixes

Resources: Links Referenced Above

- David Kolb's adult learning cycle model: http://www.simplypsychology.org/learning-kolb.html
- Stacked learning: https://www.codefellows.org/blog/how-to-accelerate-your-learning-with-stacked-modules

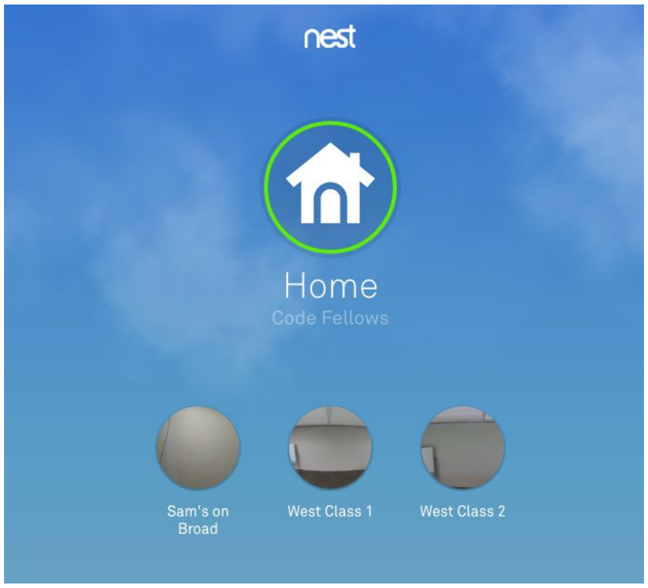
Nest Cam Video Extraction Instructions

Helvetica Neue 14pt normal font, brand blue #2662d1

Part 1 - Create Clip in Nest

1. **SIGN INTO NEST CAM** - at (https://nest.com/) using login info from PassPack - ask your instructor for the login information if you do not have it.

2. **SELECT THE APPROPRIATE CLASSROOM** - from the next screen:



3. **CREATE THE CLIP** - After selecting the appropriate camera, you will see a screen that will display either the current live stream or a message saying “Your Camera is Off”. To extract a video from the video history, click the “Video Clips” button in the bottom right-hand corner of the screen, then select “Create Clip” from the menu that pops up, see figure 2.

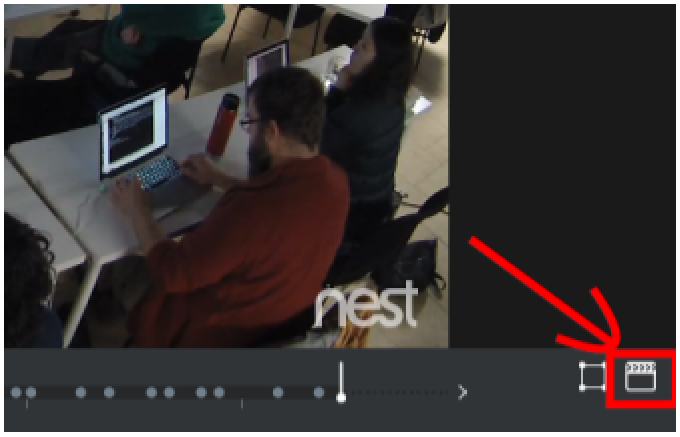


Figure 2: Create clip

4. **CREATE THE CLIP, 2** - “Create Clip” option will automatically populate a blue selection slider in the time selection bar. Move this slider around to select the

appropriate section of class that needs to be exported. The video CANNOT be over 1 hour, therefore please divide each class into 3 videos, see figure 3.

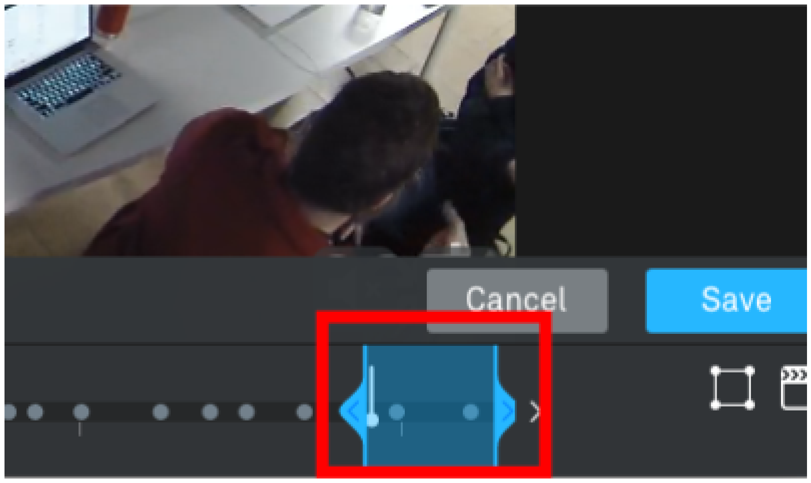


Figure 3: Selection slider

5. **CREATE A CLIP, FILTER CONTROLS** - There are time filtering controls on the bottom left-hand side of the screen - they allow for selecting the date and the duration of time that the time selection bar will display. Please ensure that the “timelapse” box is unchecked prior to saving a clip. “Timelapse” is automatically selected for clips over 1 hour, be very careful when selecting by paying attention to the time slots in the time selection bar in the yellow rectangle, see figure 4.

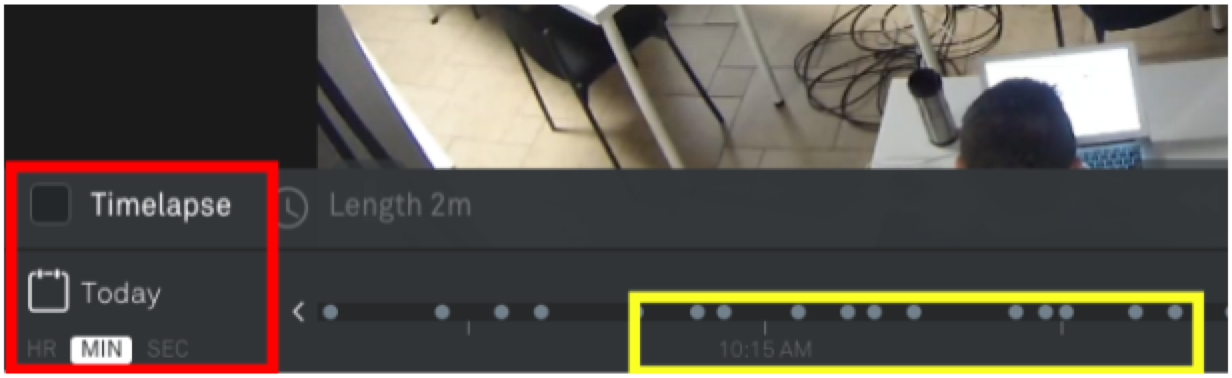


Figure 4: Filtering controls, timelapse box

6. **CREATE THE CLIP, SAVE THE CLIP** - by hitting the blue "Save" button that is located above the "Video Clips" button. This option will create a pop-up that allows for naming the clip (red rectangle, figure 5) and downloading it.

   Name the clip with the following convention "[CourseCode][Date]Part [number] of [number]", for example:

   > "sea-201d1 11.06.15 part 1 of 3"
   >
   > "sea-d45 11.06.15 part 2 of 3"

   Please download the file (yellow rectangle, figure 5) and do **NOT** save to Nest - Nest will only hold 3 hours of clips per camera. Delete the clip after downloading it (green rectangle, figure 5).

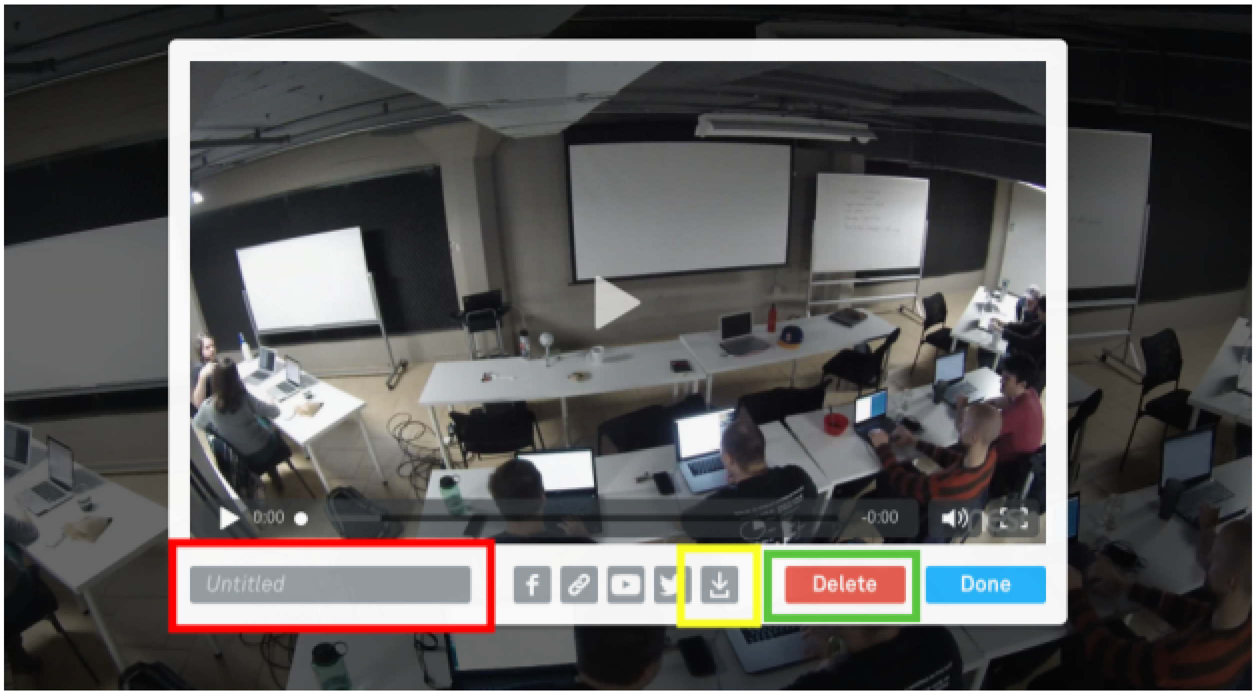


Figure 5: Saving the clip

7. **TROUBLESHOOTING** - If the application does not allow you to create a clip then select the “Video Clips” button and go to “View Clips”. Select a clip from the library by clicking on it directly then hit “Delete” in the subsequent pop-up window.

Part 2 - Upload to YouTube

8. **ACCOUNT MANAGER ACCESS** - Request account manager access for the “Code Fellows Lecture” channel from the Principal Teaching Assistant (@surfwalker (Slack) or alex@codefellows.com).
9. **ACCOUNT MANAGER ACCESS, 2** - Follow the prompts in the automated e-mail that will be generated by the system after the Principal Teaching Assistant confirms the invitation to manage the account was sent.
10. **YOUTUBE** - Go to https://www.youtube.com/, sign in using your personal account that is associated with the “Code Fellows Lectures” channel, ensuring the correct channel is selected from the account selection tool (red rectangle, figure 6) in the top right hand corner of the website, and the current account selection should appear like it does in the yellow rectangle, figure 6.

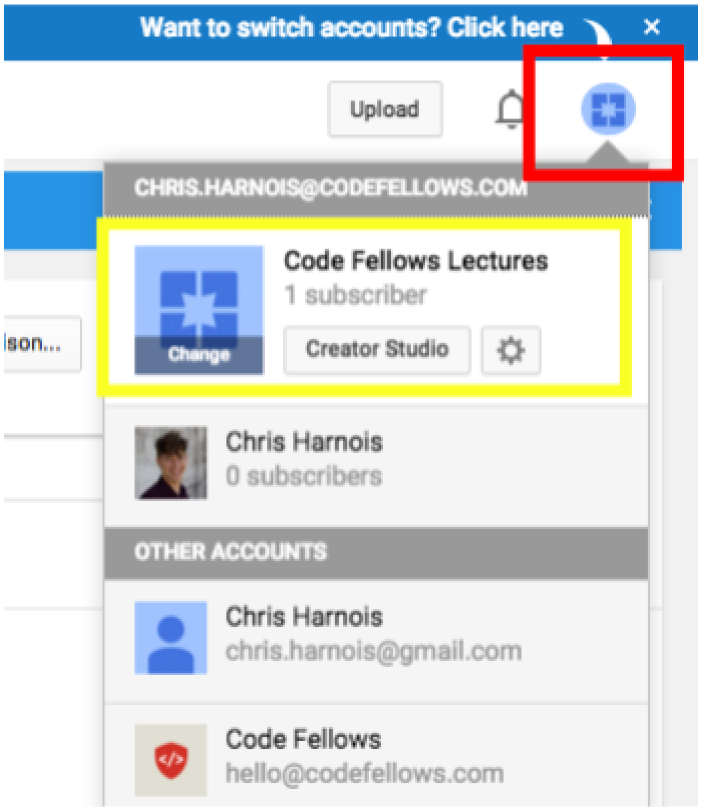


Figure 6: YouTube Account Selection

11. **YOUTUBE, UPLOAD** - Select the “Upload” button in the top, right-hand corner of YouTube to start uploading class videos. On the next screen, make sure to set the permissions of the video to “Unlisted” (red rectangle in Figure 7).

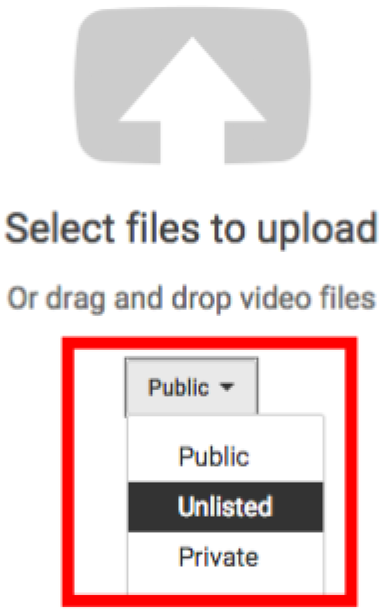


Figure 7: Privacy Settings

Code 101 Guide

Thanks in advance for helping introduce people to the world of code! Code 101 is truly an adventure - both for you, and for the students who are often going way out of their comfort zones to take this class. You are representing Code Fellows to new people for the first time, quite often. Here are some important things to know about this class, and your role in making it successful:

- Unlike some other classes, the primary goal of this class isn't learning to code. Yes, we teach students HTML and CSS, but the goal is broader: to help them determine if learning to code, and perhaps becoming a software developer, is something that's right for them. As such, we try to give students an accurate picture of what the industry is like, what kinds of jobs are out there, and what it's like to learn to code.
- These students are beginners. It sounds obvious, but there's a profound difference between true beginners and even advanced-beginners: advanced-beginners know enough to ask questions. True beginners don't even know they should have questions yet. This means that you can't wait for students to ask you for help; you'll be more effective crouching down next to someone, and saying "let me see what you have." You'll find things to help them with in no time.
- Learning to code steps on everyone's buttons at times. This means that all kinds of emotions come up in this class: anger, fear, frustration, saying "this is stupid", needing to rest on their laurels and tell you about the other things they're good at...we see it all. Instead of reacting to their anger/fear/whatever, normalize the experience of "not getting it" - let them know everyone (including yourself) has challenges, too. This is why the Code Fellows TA mantra is: "We lead with humility and empathy to create safe and inclusive environments where people learn and lives are transformed." It's up to us to create that safe and inclusive environment.
- This class isn't about learning to code, It's about trying on this field for size and making a decision about whether they want to learn more. The more you keep the focus on that, the less people are worried about not learning enough code, or getting something wrong, or not having amazing CSS, etc.

- Be really forward about the fact that the coding in this class will start at the very beginning and not assume any prior knowledge. This makes your beginners feel safer and your more advanced folks know what to expect.
- Timing: there is some play in the schedule, but if students get too far behind, they'll never recover. This is definitely one of those classes where you have to be on the ball, be careful about which stories you tell, and stay away from all those tempting diversions (penguins! Code Fellows jokes! Cool topics you like!).

Workshop Outline

TIME	TOPIC	YOUR ROLE
8:30am	Prep time	Room set-up, prepare check-in
9:00am	Intros	Be prepared to briefly introduce yourself to the class, after the instructor's intro. Questions you'll answer: you name, why you're here, and what you want to get out of the class.
9:20am	How the web works lecture	Clean up the check-in table, organize the food table (if needed), relax. Topics covered: request/response cycle, front-end, and back-end, what a "stack" is, sample web applications, the roles HTML, CSS, and JavaScript play on the front end.
10:00am	Project brainstormin g and wireframing	Student work time. Each table group comes up with a website to build today. Their site should have 1 page for each pair of people in their group (e.g. if a group has 4 people, their site has two pages; groups of 6 make 3 pages). One pair takes the home page; the other pair(s) do the other pages of the site. By the end of this time, teams should have: ● Their site idea. ● Each pair creates a wireframe(drawn on paper) for their page. ● A slack direct-message group for their team. What to watch for: ● If groups are stuck thinking of an idea, it can help to ask them to each share what they like to do for fun, or community groups they're involved in. ● Have them "pitch" their idea to you: what the idea is, what

		each pair's page is (they often get confused on how many pages they're building). ● Coach them towards as detailed a wireframe as possible. Grouping content into blocks now will make the HTML section easier. ● Check that each student is in the Slack channel.
10:30am	HTML lecture/ code demo	Topics covered: the role of HTML (content, and organizing that content); HTML syntax; paragraphs, lists, images, links, and some HTML semantic tags. Also: how to use an editor, how to view that HTML in the browser.
11:15am	HTML work time	Pairs code the HTML for the wireframe they created earlier. In their pairs, each person is writing the HTML on their own computers, but they're talking back and forth and helping each other debug. What to watch for: ● All content should be in one of four organizing tags: a <header>, a <nav>, a <section>, or a <footer>. You can refer students to the instructor's code for an example; there will be a link to it in Slack. Students generally struggle with how to group and organize content. ● Indentation. If they can have cleaner indentation earlier on, it makes debugging easier. ● File names: as they make their links for their nav bar, they should ask the other members of their team what filenames they are using for their pages, and use those filenames. ● File path: check that their files are in a path that doesn't have spaces in it. Same goes for filename. Not urgent now, but it sets them up for success when we get to the terminal. ● Whichever pair has the home page should have their HTML file named "index.html" (no other names, no capital letters). Again, not urgent now, but sets them up for success with GitHub pages.
12:00pm	Lunch	Eat some lunch. Some students will elect to keep working on their HTML during this time; help them if need be.
12:30pm	Q&A	

12:45pm	CSS lecture	Topics covered: role of CSS, linking CSS files to HTML; selectors and properties; changing font sizes/colors, box model, classes (if time), how to use documentation to look things up.
1:30pm	CSS work time	Pairs create CSS files based off a template, then edit properties as a way to get started. Eventually, something will pique their interest, and they'll figure out how to implement properties of their choice. What to watch for: ● This section is intentionally more open-ended than the HTML section. Students start with the template, but then branch off into different topics of their choosing. We fully expect that they'll break things, get frustrated, have to ask for help, and (hopefully) figure some things out in the end. This is more like the actual experience of learning to code, and of being a developer in real life. ● Helping students is a mix of direct instruction ("here, try this") and telling them about a concept (e.g. background-image) and then helping them look it up. ● Within pairs, it's normal for one student's code to end up different than their partner's, especially if one has prior experience with CSS.
2:15pm	What it's like to be a developer lecture	Take a break!
3:45pm	Assemble sites	The goal for this time: Each student has their page uploaded to a repl and the links to the other team members pages are working. We need to check off the following for each student: ● No spaces in filenames ● No spaces in their path ● Site works (mostly) locally on their computer OR at the very least they have an index.html ● Home page is named index.html (yes, this is case sensitive) In general, it helps to reassure students that this doesn't have to be the "final" version of their site.

4:15pm	Evaluations/ prepare presentation s	Students complete the evaluation of the class. The team should also verify that their site is working properly (links to each team members page work). One team member then needs to share the link to the home page in the Slack channel. ● Check in with each student (individually works better than with the group all at once) at your assigned table(s) and make sure they got the link. Remind them to do the survey. ● Make sure they have a link in Slack. If there's time, or the class is small, we may ask them to prepare a 3-5 minute presentation about their site.
4:30pm	Present sites	Each team presents their sites. If the class is large, this is often mediated by the instructor; if small, the groups have more time to present. Cheer loudly!
5:00pm	Clean up and debrief	Clean up the room; debrief the class; head home!