Guide to Differentiation

CS Discoveries Approach to Differentiation

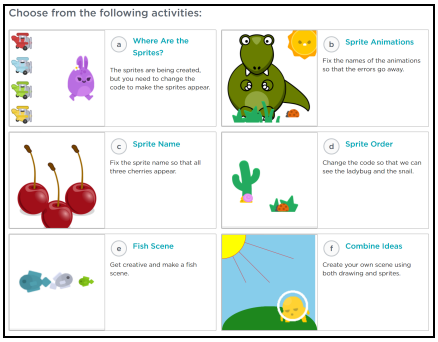
In order to meet the needs of a wide variety of learners, CS Discoveries is designed with flexibility that allows teachers to differentiate their instruction at the class and student level. Students are expected to take an active part in driving these learning experiences, seeking out appropriate levels of challenge and support with the guidance of a teacher. The course includes multiple features that allow teachers to differentiate instruction based on the levels and the needs of their students while ensuring that the class stays together and maintains a sense of community.

Three Ways to Differentiate Lessons

Differentiation is an essential tool for achieving equitable access in your computer science classroom, where every student regardless of race, gender, language, ability, or socioeconomic background has an equal opportunity to succeed. CS Discoveries lessons and projects are designed to meet these various needs of students within a classroom cohort while keeping them moving at the same pace throughout the course. To that end, below is a summary of the different ways with examples that you could differentiate instruction in your CS Discoveries class.

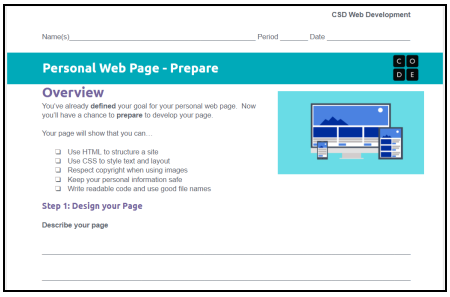
Differentiation by Task

Differentiated tasks allow students to practice a new skill or demonstrate mastery of a concept while providing flexibility in the product or program they create. This can look like offering students choice in the task they complete, where varied tasks appeal to different interests or skill levels. This can also look like "low flow, high ceiling" tasks, whose initial requirements are achievable by all but have a variety of options for how students can extend and finalize their program. With this style of differentiation, all students in your classroom cohort are working on the same type of assignment or project, but the choices they make within the project are what lead to differentiation.



Example: Practice Levels. Practice levels can be found in lessons after students have been introduced to the target skills before the assessment level. Practice levels provide multiple ways to engage with the lesson content, including debugging activities and modifying existing code.

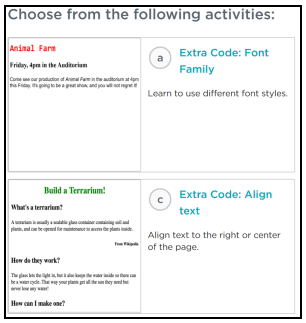
With the guidance of a teacher, students may choose to complete all the practice options or just as many as they need to be ready for the assessment. None of the practice options introduce new skills or concepts, so students who skip ahead to the assessment level will not have missed any key material. Previews and short descriptions help students to find an activity that appeals to them.



Example: Projects. Open-ended projects are found throughout the programming units, with a major project at the end of each chapter. While each project includes a rubric that assesses key learning objectives, students may choose the content of their projects, and teachers may choose to add or alter requirements for a class or individual students. These requirements could include extra challenges for those who should go further, or they could provide scaffolding for students who need more guidance.

Within each project lesson, students will be asked to plan out their project before starting to code. Teachers may want to look over these plans and ensure that students are choosing a level of difficulty that is appropriate for them. As students iterate on their projects, teachers may want to provide extra challenges or help them to modify their plans to make them more manageable.

Example: Challenge Levels. Challenge levels are found after the assessment levels in many of the programming lessons. These levels include new code and challenges that go beyond the learning objectives of the lesson. Most also include a "Free Play" option that allows students to use the new skills they have learned in whatever way they choose.



Although these levels include new content, students will be able to move on to later lessons without needing anything introduced here, and none of it is necessary to achieve the goals of the course. These levels provide students who have already mastered the lesson content with new challenges and extension code, giving them more options to express themselves creatively in the programming environment. As those students take on these more difficult activities, teachers can provide targeted support to students who are still working toward mastery of the lesson objectives.

Differentiation by Timing

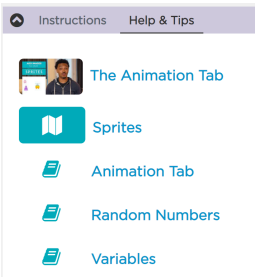
For some tasks, you may find students need differing amounts of time to complete tasks. This could be because a student needs additional time and support to master a concept, or because a student has given themselves a challenge and would like to see their idea through to completion. In either case, being flexible with deadlines helping students set individual goals for their work can help meet the unique needs of each student.

Example - Mini Projects. Mini-projects are smaller projects that appear throughout CS Discoveries units. These projects act as checkpoints in the curricula and cover a smaller set of skills than the entire unit. Though they are only scoped for a single day, teachers may decide to extend these projects as a way to support or challenge students. Extending these projects may allow you to revisit difficult concepts or support students who may have missed lessons and are trying to catch up.

Example - Implementation Options. Although the course comes with a standard pacing guide, class schedules and student needs will make modifications necessary for most classrooms. Teachers should feel free to spend more time where students need it, even if it means skipping other parts of the course. In particular, the first chapter of every unit ends in a project that will wrap the content up nicely if there is no time for a second chapter.

Differentiation by Resources

Learning to use resources is a key goal of the course, and given resources provide an opportunity for students to self-differentiate in how they interact with key course content. This may include proactive differentiation, such as printing out resources ahead of time for students. It can also include just-in-time differentiation, such as monitoring students as they reach the end of a project and referring them to additional resources to extend and challenge themselves.



Example - Help and Tips. Programming levels include a "Help and Tips" tab that includes concept explanations, code documentation, and videos to support students in completing their activities. Resources can be printed ahead of time and made available to students who would benefit from a tangible document that supports their task in Code Studio. We also have a guide to using resources that provides additional tips for supporting students with resources.

Example - Documentation. Each code block includes a link to "See Examples", which opens the documentation for students. Students can see examples of how the block behaves and a written explanation of how to use the block in their program. Students can always access this documentation if they need a reminder on how a block works. Or, students can get inspired by the examples as an opportunity to extend their programs and challenge themselves.