Unit 3 - Intro to App Design

Unit Overview

Students design their first app while learning both fundamental programming concepts and collaborative software development processes. Students work with partners to develop a simple app that teaches classmates about a topic of personal interest. Throughout the unit, they learn how to use Code.org's programming environment, App Lab, to design user interfaces and write simple event-driven programs. Along the way, students learn practices like debugging, pair programming, and collecting and responding to feedback, which they will be able to use throughout the course as they build ever more complex projects. The unit concludes with students sharing the apps they develop with their classmates.

Unit Philosophy and Pedagogy

- **New Topics, Same Classroom Culture:** This unit is students' first experience with programming but it is designed to maintain the collaborative and inclusive classroom environment developed in the previous two units. The collaborative project, fun unplugged activities, and the focus on experimenting should help keep your whole class working together and trying out ideas.
- **Emphasizing Skills:** Since this is the first of many programming units, it emphasizes attitudes and skills that will serve your students well for the remainder of the year. The project that runs through this unit emphasizes that programming is a creative and collaborative endeavor that can be used to help others. Key practices like pair programming and debugging help normalize working with a partner, asking for help, and making mistakes. While students have a lot to learn about programming and App Lab, there is just as much emphasis on establishing these positive habits and mindsets.
- **Empowering "Creators":** This unit empowers students to be creators with a major emphasis on making projects that are personally meaningful. Students have a lot to learn about programming, but the goal is that they come away from this unit seeing programming as a powerful form of personal expression that allows them to draw on their innate talents and interests to help solve problems in their community.

Major Assessment and Projects

The unit project asks students to collaborate with a classmate to design an app that can teach others about a topic of shared interest. Students practice interviewing classmates to identify the goals of the project, mock up designs, collaboratively program the app, and run simple user tests. The app itself must include at least three screens and demonstrate what students have learned about user interface design and event-driven programming. Students submit their app, project guide, and written responses to reflection questions about how the app is designed and the development process used to make it. Students will also complete an end-of-unit assessment aligned with CS Principles framework objectives covered in this unit.

AP Connections

This unit and unit project helps build towards the enduring understandings listed below. For a detailed mapping of units to Learning Objectives and EKs please see the "Standards" page for this unit.

- CRD-1: incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- CRD-2: developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- AAP-2: The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle

varied input values.
- AAP-3: Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.

This unit includes content from the following topics from the AP CS Principles Framework. For more detailed information on topic coverage in the course review **Code.org CSP Topic Coverage**.

- 1.1 Collaboration
- 1.2 Program Function and Purpose
- 1.3 Program Design and Development

The College Board has supplied formative Create PT questions to help prepare students to complete the Create Task. We recommend that students complete the following prompts with the unit project. More information can be found in Code.org CS Principles Topic Coverage.

- 3.a.i.
- 3.a.ii.
- 3.a.iii

Week 1

Lesson 1: Introduction to Apps

App Lab

Welcome to the Introduction to App Development! We begin this unit by examining the basics of an app.

Lesson 2: Introduction to Design Mode

App Lab

Learn how to create screens, buttons, images, text, and more inside of App Lab.

Lesson 3: Project - Designing an App Part 1

Project

This is the first in a series of lessons where students will make progress on building their own functional app.

Lesson 4: Project - Designing an App Part 2

Project | App Lab

Students continue working on the unit projects in this lesson that is primarily designed to be work time. Students continue to follow the app development process outlined in their App Development Guide by transferring their user interfaces designs from their planning guides over to App Lab

Lesson 5: The Need for Programming Languages

This is the first in a series of lessons where students will make progress on building their own functional app.

Week 2

Lesson 6: Intro to Programming

App Lab

Get familiar with what programs are and how they run by using and modifying several working apps.

Lesson 7: Debugging

App Lab

Learn how to identify and fix problems in your code through the debugging process. Then practice using it with a classmate.

Lesson 8: Project - Designing an App Part 3

Project | App Lab

Start programming your app and learn how to effectively use pair programming to collaborate while writing code.

Lesson 9: Project - Designing an App Part 4

Project | App Lab

Continue working on your app and get feedback to make sure you're on the right track.

Lesson 10: Project - Designing an App Part 5

Project | App Lab

Complete your app and share it with classmates before reflecting on the overall process of designing the app.

Week 3

Lesson 11: Assessment Day

Project

Complete your app and share it with classmates before reflecting on the overall process of designing the app.

Lesson 1: Introduction to Apps

Overview

Students explore and investigate what makes an app an app. They begin by looking at and discussing five different apps. Following this, students watch a video explaining the basics of how computers work. Finally students return to the apps and consider the various inputs and outputs.

Purpose

This lesson is an introduction to a unit that introduces programming through the broader context of designing apps that help people. This lesson is designed to build excitement about the skills that students will develop throughout the unit and get them thinking early about the ways programming can help others. This lesson also establishes shared vocabulary for talking about apps in a generic sense, for example its inputs, outputs, and overall user interface.

Agenda

Warm Up (5 mins)

- **Preview the Unit**

Activity (30 mins)

- **App Exploration**
- **Input & Output**
- **App Investigation**

Wrap Up (10 mins)

- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Identify the inputs of an app
- Identify the outputs of an app
- Identify the purpose of an app

Preparation

- [ ] Preview the How Computers Work Video
- [ ] Explore the apps used in this lesson

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **How Computers Work - What Makes a Computer, a Computer** - Video

Teaching Guide

Warm Up (5 mins)

Preview the Unit

Remarks

> Now that we've learned how to represent information digitally and how the Internet works, it's time to think about creating digital content for others to interact with.

Prompt: What are apps? How do we interact with them? What kinds of things do apps do?

Discussion Goal

Apps are designed to solve a problem by providing a service

- Entertaiment: images, movies, tv shows, etc.
- Social: social media, chatting
- Commerce: marketplaces
- Finding Information: search engines, maps

Apps have screens or *User Interfaces* to display and collect information to and from users

- These things collect information: Buttons, sliders, text boxes, etc.
- This is the type of information that can be displayed: Sounds, images, video, text

Activity (30 mins)

App Exploration

Remarks

> Today we are going to be looking at some sample apps to explore their purpose and function.

Group: Organize students in groups of two. Students should navigate to the App Exploration starting on level 2.

Levels 2-6: There are five different sample apps. Depending on time, students can explore 3-5 different apps.

Prompt: With a partner, discuss the following and note down in your journal:

- How does the user interact with the app?
- What is the overall purpose of this app?
- Who is the target audience?

Share Out: As a class, discuss student answers to the discussion questions.

Remarks

> In developing apps, which are a kind of computing innovation, it's important to understand both their purpose and function. The purpose helps the developer focus on creating an app with specific goals in mind.
>
> Now let's dig in to how the user interacts with the apps.

Input & Output

Display: How Computers Work **How Computers Work - What Makes a Computer, a Computer - Video**

Remarks

> The video explained *Input*, *Storage*, *Processing*, and *Output*. Today we are going to focus on input and output. An app on a phone gets input in may different ways from a user or from another program and displays (or in the case of sound: plays) output. The output of a program is usually based on inputs or the initial way the program was set up. For example, you could program an app to play a song as soon as it runs.

App Investigation

Levels 8-12: With your partner, take another look at the sample apps you explored before by navigating to the App Investigations starting at level 7. Consider what the inputs and outputs are for the apps. Note these down in your journal.

Wrap Up (10 mins)

Remarks

> Apps solve a problem for a target audience. Apps take in input from users and output information in various ways. Users interact with apps through the *user interface*.

Prompt: Think of your favorite app. Discuss with a partner what the user interface looks like and the inputs and outputs.

Journal Add to your journal the following terms along with your own definitions:

- User Interface
- Input
- Output

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Match term with the example.

Discussion Goal

App #1: Water Conservation Tips

- The user clicks through the different screens and makes choices about the correct ways to conserve water. When the user makes choices, images or sounds are played to indicate if a choice is the right one.
- The purpose of the app is to teach people about ways to conserve water.
- The target audience is young people who are interested in the topic

App #2: Bird Quiz

- The user answers trivia questions about birds. If the answer is correct, they go to the next question. If wrong, they are taken to a losing screen, and then start the quiz again. If three questions are answered correctly the user is taken to the winning screen.
- The purpose of the app is to teach users different facts about birds
- The target audience is anyone interested in birds

App #3: Hamilton Township

- The user clicks through a few different screens to learn about township efforts to beautify the town
- The purpose of the app is to educate users on steps the town is taking to improve the town
- The target audience is people who live in Hamilton Township

App #4: Four Square Instructions

- The user navigates through different screens that explain how to play four square. On one screen, the user is quizzed on how many people are needed to play four square.
- The purpose of the app is to instruct the user on the rules of four square.
- The target audience is people who would like to play four square.

App #5: Monarch Butterflies

- The user clicks on different screens and pictures of monarch butterflies to explore the life stages of the butterfly
- The purpose of the app is to educate users on the life cycle of the monarch butterfly
- The target audience is young people, most likely children or teenagers who are interested in life science and butterflies

Teaching Tip

After students are finished writing in their journals, discuss as a class or collect the journals to review student answers.

- An input could be a user clicking a button or tapping the screen
- An output could be an image displayed or a sound played

Discussion Goal

After students have shared with a partner, bring the class back together and discuss a few examples.

Standards Alignment

CSP2021

▶ **CRD-2** - Developers create and innovate using an iterative design process

Lesson 2: Introduction to Design Mode

Overview

Students work through a progression of levels to build an understanding of how to use Design Mode to layout an app. The final level has students setting up the screen of an app by attempting to copy an image of an app.

Purpose

This lesson introduces students to Design Mode in App Lab and the different kinds of screen elements and properties at their disposal in this tool. This lesson not only builds up skills in designing a user interface but also sets students up to begin designing an app of their own in the following lesson.

Agenda

Warm Up (5 mins)
Activity (30 mins)
Wrap Up (10 mins)
Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Use meaningful names to for element ids
- Set up the User Interface of an app including buttons, text, and images

Preparation

- [ ] Review the programming levels to ensure you understand the fundamentals of Design Mode in App Lab and how elements are created and their properties are modified.

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

Teaching Guide

Warm Up (5 mins)

Prompt: What is a common app that you use? Take a minute to sketch the User Interface of the main screen. Note how the user interacts with the app.

- Compare your sketch with a partner and discuss common elements that both of your apps share

Remarks

> Yesterday we took a look at a few different apps and talked about input and output. We also discussed the User Interface. Today we are going to learn how to build the User Interface in App Lab.

Discussion Goal

Goal: This prompt is designed to bring out students' prior experiences working with apps. If students aren't sure what app to choose, you can even ask them to pick to Code.org website. This discussion can pull out the fact that most apps contain images, buttons, and text that help users find and interact with information and content.

Activity (30 mins)

Do This: Direct students to Code Studio.

Level 2: Students examine an app from the previous lesson. They should feel free to move around and change any elements they'd like. Encourage students to spend a good amount of time on this level discovering how things work. Here are a few things they should notice:

Teaching Tip

Regrouping: You may want to pull the class back together after Level 2 to discuss their findings.

- The screen can be changed by clicking the dropdown at the top of the app screen
- Each element has different properties that can be changed depending on the type of element. For example, a label element has choices like "text alignment" and "font family" while an image has choices like "fit image".

Levels 3-7: In these levels, students will learn how specific elements are set up and edited. Level 3 introduces the concept of themes, which control the overall "look" of an app. No matter what theme is chosen, students have the ability to override any element's design.

Level 8: Students work from a given image to recreate the user interface for an app. It's ok if their design does not exactly match the sample. The goal is for students to become comfortable setting up a User Interface before the sketch out plans for their own apps in the next lesson.

Wrap Up (10 mins)

Prompt: Have students answer the following

- What elements collect input?
- What elements display output?
- Do you think there are elements that can do both?

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Why is it important for element IDs to have meaningful names?

Discussion Goal

What elements collect input?

- Buttons
- Text Input
- Check Boxes & Radio Buttons
- Dropdown

What elements display output?

- Images
- Text

Do you think there are any elements that can do both?

- This is a tricky question! Most element can do both, because they can all be clicked and can change their content. Students will see this in action in the next few lessons.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Lesson 3: Project - Designing an App Part 1

Overview

This is the first in a series of lessons where students will make progress on building their own functional app. In this lesson, students brainstorm app ideas and sketch out user interfaces in preporation for the next lesson where they will return to App Lab.

Purpose

This lesson kicks off a project that students will complete throughout the unit. The framing of this project is also important for how programming is presented overall. Students are encouraged to collaboratively design their projects, choose topics of personal interest, and build an app to meet the needs of other people. All of this is important as part of framing programming as a collaborative, creative, and socially situated pursuit.

Agenda

Warm Up (5 mins)
Activity (30 mins)
Wrap Up (10 mins)
Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Brainstorm and choose a topic to develop into an app
- Use feedback to help guide the design of an app
- Design the user interface of an app

Preparation

- [ ] Read over the **App Development Planning Guide - Activity Guide**

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

Prompt: People design user interfaces to meet a user's needs, but they don't always get it right.

- Have you ever used an app where the user interface didn't actually meet your needs?
- What was the problem?
- What do you think the designers didn't understand about you or your needs?

Discussion Goal

Goal: The goal here is to help students understand the importance of considering the end user when designing an app.

Some points that may come up:

- Some apps have too much guidance on how to use them, and others too little.
- An app may be designed for a phone, but the user wants to use it on a tablet.
- Apps that are designed for teenagers may make assumptions about language usage or interests that aren't accurate.

Activity (30 mins)

Remarks

> Yesterday we learned all about Design Mode. Today you're going to start developing your own app that educates a user on a topic. This is a project that will continue through the rest of the unit.

Distribute: App Development Planning Guide - Activity Guide. Students will be using the Planning Guide for the rest of the unit. Read the Project Description together and make sure students understand the requirements. Direct students to the "Investigate" section.

Step 1: Brainstorm Topic Ideas

- Students have a lot of freedom in choosing their topics. If they are struggling with ideas, they could create an app for another class, for example: an overview of the periodic table for science class.

Group: Organize students into pairs.

Remarks

> You will work with a partner on this project. Collaboration is a big part of app development. Your team's diverse ideas will help make your app even better, because each partner can keep a lookout for bias.
>
> Here's an example: A team member puts in their app the phrase: "All teenagers work after school to earn spending money" because this is their experience - they work after school themselves. However, the other team member may not have an after school job, and because of this can point out the bias in that statement.
>
> Collaboration is key to catching bias in your work!
>
> Here's another example, this time in the design of an app. A team wants to make an app to encourage girls to sign up for a summer camp. One team member assumes that because this app is aimed at girls, the background should be pink. A girl on the team points out that she does not like pink, and using that color in the design may represent bias.
>
> As you think about your topics and start designing your apps, remember to keep an eye out for bias and use your team mates to keep you accountable!

Step 2: Choose One Topic

- Working with a partner students narrow down their ideas to one topic and explain what would be covered in their app.

Step 3: Survey Your Classmates

- The goal in this step is for students to understand the needs of their users. This will help inform the overall design of the app.

Step 4: Design the User Interface

- Student use the screen templates to sketch out the design of their app, including arrows or notes to show how different elements are connected.

Steps 5+ will be completed later on in the unit.

Teaching Tip

After Step 4, you may want to check in with student groups to make sure their apps are doable and can be completed within the scope of the project.

Wrap Up (10 mins)

Prompt: How did talking with the users of your app impact your design decisions?

Remarks

In the following lessons, you will continue to plan and create you apps. What are we doing here? We are following a development process, which is similar to what developers use in the real world when creating their own apps. A development process breaks down the steps into small pieces - we've completed the first few today.

There are many different version sof the development process. Most use some combination of investigating and reflecting, designing, prototyping and testing. Development processes can be ordered like we have layed out in the Activity Guide or more exploratory in nature. As we continue on through this project, reflect on the steps and consider how they help organize the process.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: Match the term with the examples.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **CRD-1** - Incorporating multiple perspectives
- **CRD-2** - Developers create and innovate using an iterative design process

Lesson 4: Project - Designing an App Part 2

Overview

Students continue working on the unit projects in this lesson that is primarily designed to be work time. Students continue to follow the app development process outlined in their App Development Guide by transferring their user interfaces designs from their planning guides over to App Lab

Agenda

Warm Up (5 mins)
Activity (30 mins)
Wrap Up (10 mins)
Assessment: Planning Guide

View on Code Studio

Objectives

Students will be able to:

- Create a user interface based on a prototype

Preparation

- [ ] Make sure students have access to their App Development Planning Guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

Remarks

> In the last lesson, you developed a prototype of an app with a partner. It's important to remember that other people will be using your app and need to understand how to use it!

Prompt: Why is it important to plan out the design of an app?

Discussion Goal

- Planning can save time so the developers don't spend time designing faulty user interfaces
- Planning can help the developers talk through and test out ideas with the users of the app

Activity (30 mins)

Remarks

> The design you sketched out is a prototype of your app. A prototype helps you plan the looks and features of your app before you program it."

Do This: Direct students to Code Studio.

Level 2: On this level, student see a sample app they've looked at before along with the sketches for the screens. Students run the app and compare the screens to the original design.

Remarks

> After looking at this app and the prototype, you may want to make a few adjustmants to your own prototypes. Take a few minutes to update your design.
>
> When you are finished, move on to Level 3 where you will start creating your user interfaces in App Lab!

Level 3: Here students will work on transferring their prototypes to screens in App Lab. Partners can work together sharing a computer, or they can divide the screens between themselves and work individually. After they are done, the individual screens can be shared to create a complete project.

Teaching Tip

Students should explore the app and the prototype for a few minutes, noting how different elements are represented on the prototype.

Encourage Collaboration: Reinforce a collaborative classroom environment in which students share responsibilities for designing screens and discuss with one another the choices they're making. Students should be communicating their ideas with one another as they build.

Waiting for Programming: Some students may be eager to start programming their apps rather than simply build screens. Let students know that they'll be able to start programming their apps in the coming lessons and that the first half of the unit focuses more heavily on design.

Wrap Up (10 mins)

Prompt: Were there any changes you had to make to your original design once you transferred it to a screen?

Assessment: Planning Guide

This is a good opportunity to look over students work in the **App Development Planning Guide - Activity Guide** as a formative assessment.

Teaching Tip

The Wrap-up is short today to allow students the maximum amount of time to work on their apps.

Often times students will discover that their design is too complicated on paper and simplify it when they transfer it to the screen. This is ok! Sometimes the best designs are simple designs.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

Lesson 5: The Need for Programming Languages

Overview

In this lesson students explore the challenges of clearly communicating instructions. They build a small arrangement of blocks (LEGO® pieces or paper cutouts) and then create text instructions a classmate could follow to construct the same arrangement. Groups then trade instructions to see if they were clear enough to allow reconstruction of the original arrangement. The wrap-up discussion is used to highlight the inherent ambiguities of human language and call out the need for the creation of a programming language which leaves no room for interpretation.

Purpose

This lesson is students' first introduction to the concept of a programming language. It helps them understand why programming languages exist by giving them a first-hand experience with the problems that programming languages are designed to address. Natural language is usually too ambiguous for giving precise instructions that can be followed correctly 100% of the time. We need to create more structured and precise programming languages in order to accomplish this.

Agenda

Warm Up (5 mins)
Activity (30 mins)
Wrap Up (10 mins)
Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Justify the existence of programming languages to precisely communicate instructions
- Explain the qualities that differentiate natural languages and programming languages

Preparation

- [ ] Prepare either a small set of LEGO blocks or paper cutouts for each pair of students

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **The Need for Programming Languages** - Presentation

Teaching Guide

Warm Up (5 mins)

Remarks

> Today we're going to look at what it takes to write "good" instructions. We all use instructions all the time, whether it's directions to get somewhere, instructions to fill out a form, or even instructions from teachers. To help us get started, let's brainstorm what "bad" instructions" look like.

Prompt: Write down three different reasons you would call a set of instructions "bad". Be ready to share with a neighbor.

Discussion Goal

Goal: Aim to hear a few different students share reasons that instructions are "bad". The point here is just to get students thinking and there's no specific answer you're driving towards. Some possible ideas, however, might include:

- Instructions are not clear on what to do
- Instructions use confusing words
- Instructions don't actually accomplish what they're supposed to

Remarks

> Alright, so we have an idea of what "bad" instructions look like. Let's see if we can use this brainstorm to help us write good instructions in the next activity!

Activity (30 mins)

Group: Place students in groups of two, optionally forming a group of three.

Distribute: Give each group a small set of blocks (either LEGO or paper cutouts). Also have each group use a blank sheet of paper or a blank page in a journal.

Display: Use the lesson slides to guide this activity.

Step 1: Design: Give students a couple minutes to create their design. Keep this quick since the bigger focus should be on the instructions.

Step 2: Record: Either have students sketch their design or take a picture. Make sure the image is on a separate piece of paper or the back side of a sheet of paper.

Step 3: Write Instructions: This is the most important step and should be given about 5 minutes. Encourage students to be as clear as possible while only using words. Encourage them to think about their brainstorm of "bad" instructions this morning to see if it can inspire them to make "good" ones.

Step 4: Trade: Have students take apart their design, and then trade instructions with another group. Make sure they keep their recording of the design hidden.

Teaching Tip

Make it a Gallery Walk: Depending on your room set-up you may be able to have students run this activity in a more open-ended way. Students should leave their unconstructed blocks and instructions at their table with the target image face down. They can then make their way to a few different groups' instructions over the course of 10-15 minutes.

Mark What's Confusing: You may optionally have students leave suggestions on instructions when they use them explaining where they were confused or wanted more detail. This is a good way to practice giving and receiving feedback. It also forces groups to think more clearly about what is causing their confusion since they'll have to leave it in writing.

Step 5: Build: Give students 3 minutes to follow the instructions from another group.

Step 6: Compare: Give students 2 minutes to compare what they built with the picture of the actual target. In their groups they should discuss what they think went wrong or anything they're surprised worked out.

Step 7: Repeat: Depending on how much time you have, encourage students to trade with one or two other groups. Make sure to save time for the wrap up.

Wrap Up (10 mins)

Prompt: When you or your classmates made mistakes following instructions today what "went wrong"? Try to be as specific as possible.

Remarks

> Even trying to be as clear as possible we struggled to write clear instructions. There are things we could do to improve, but the core challenge here is that everyday human language is bad for giving clear instructions. Words can have two or three meanings! Instructions that seem clear turn out to be vague when we actually go follow them. If we want to give clear instructions, we need to fundamentally change the language we use. We need to create a new kind of language.

Prompt: Imagine we were going to redesign human language to be really good for giving clear instructions. What types of changes would we need to make?

Remarks

> Today we talked a lot about instructions because soon we're going to be start programming your apps. When you write a program you're just giving instructions to a computer for what it should do to run your app. As you'll see, the programming languages we use to give these instructions sometimes looks like English, but then has a lot of weird (and sometimes confusing) differences. This is because it needs to be more precise and unambiguous than normal human language. We'll dive into this more next time we meet!

Discussion Goal

Goal: This wrap up includes two sets of prompts and should be run as back-to-back discussions. The first set helps students synthesize the challenges they encountered during the main activity. The second set prompts students to think about how they might design a new language that avoids these challenges.

When running the first discussion ask students to share specific experiences during the lesson and prompt them to think about what went wrong. They might say things like "if we'd just been more clear" or "we just need to include more detail". This might be true, but eventually you should point out that human language is often ambiguous. In other words, it's not their fault! It's hard to do a good job with bad tools!

When running the second discussion push students to think about what a language designed for giving instructions would look like. While specific answers aren't important, you should push them to build on their experiences in the main activity. For example, if they saw that words having two meanings was confusing in the main activity, they should suggest redesigning the language so each word has a single well-understood meaning.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

Question: What is the difference between a programming language and natural (every-day) language?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 6: Intro to Programming

Overview

Students use and modify a series of simple apps to get familiar with a small set of programming commands. They observe the way the code runs by slowing down the code and compare programs that run all at once to those that respond to user actions like buttons cliks. At the end of the lesson students discuss what they observed and are introduced to some key vocabulary for describing the running of programs.

Purpose

This lesson is written in an Investigate style, a common format of lesson that will be used in the remainder of the programming units. In this kind of lesson students are encouraged to investigate working code and make simple modifications to understand how it works.

This lesson introduces a number of concepts and vocabulary around what programs are and how they run that will need to be reinforced in future lessons.

Agenda

Warm Up (0 mins)

Get to the Activity

Activity (30 mins)

Wrap Up (15 mins)

Assessment: Check For Understanding

View on Code Studio

Objectives

Students will be able to:

- Define a program as a sequence of commands that are executed or run by a computer
- Explain the differences between how sequential and event-driven programs execute
- Define comments as notes or documentation into a program that do not affect how the program executes

Preparation

- [ ] Review the example apps and the prompts that students will be asked to respond to for each
- [ ] Review the information covered in the slides at the conclusion of the lesson

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

Teaching Guide

Warm Up (0 mins)

Get to the Activity

Remarks

> Yesterday we learned that we need to create a new kind of language, a programming language, in order to clearly communicate instructions. Today is a big day. We're going to get our first taste of what programming looks like, and we'll see different ways we can use these tools to communicate our instructions to a computer.

Teaching Tip

Get to the Activity: There's a lot to do in this lesson. Get to the main activity as quickly as you can.

Prompt Ideas: If you need a prompt for when students come in consider asking them to list 2-3 differences between a natural language and programming language and why those differences need to exist.

Activity (30 mins)

Group: Place students in pairs. One student per group should navigate to the lesson on Code Studio.

Display: If possible in your room, display the Code Studio levels at the front of the class.

Level 2: Have students work on the three tasks for a few minutes. Circulate the room making sure that pairs are actually discussing the prompt and are collaborating to modify the program. Once it seems all groups have had a chance to do this bring the class together.

Discuss: Have students share the results of their discussions with the class. You do not need to use formal vocabulary yet, but make sure all students are seeing the same things.

- Code is running one line at a time.
- Strings need to go in quotes. Numbers do not need to.
- Yellow highlighting shows you which line of code is running in either block or text mode.
- The turtle slider changes the speed at which the code runs. At full speed (all the way to the right) there is no longer any highlighting.

Level 3: Run this level in the same way. Have students complete all three prompts and then bring the class together after a few minutes for a discussion.

Discuss: Once again have students share the results of their discussions and modifications with the room. Here are some good points to draw out if they don't come up naturally in discussion.

Teaching Tip

Prepping for Investigate Lessons: The best way to prepare for this lesson is to go through the experience yourself. Check out the three apps in Code Studio to get a sense for how they work. Then move on to the Code Investigation and actually try to answer all the questions for each app. To help you out, however, answers are provided on the bottom of the instructions area for verified teachers.

Show Code at the Front: If your room allows it, display the code at the front of the room. When students mentions specific lines of code actually scroll to that line and read through it together.

Discuss specific lines of code: As you run discussions, model talking about programs by specifically calling out lines of code, as in "I can see that when the playSound block on line 2 is highlighted the sound plays...."

Save Modifications for the End: This lesson can be tight on time. Rather than have students modify the code all at once, you can save modifications for the end of the Code Investigation and have students pick a single app they wish to modify.

OK to Break Things: When using the widgets in Units 1 and 2 it's not really possible to "break things". That's a little different than how things work in App Lab where it is possible to write code that may not run at all. Encourage students that this is ok. Using blocks makes it easier to avoid errors, and if students need to they can use the version history to set the code back to its original state.

- `console.log` prints text in the Debug Console
- `setProperty` changes the properties of elements on the screen. These should be familiar from using Design Mode.
- `console.log` needs one input while `setProperty` needs three

- Hovering over a block in the toolbox helps you know what kind of information to put in each input.

Level 4: Run this level in the same way. Have students complete all three prompts and then bring the class together after a few minutes for a discussion.

Discuss: Once again have students share the results of their discussions and modifications with the room. Here are some good points to draw out if they don't come up naturally in discussion.

- `onEvent` makes the program respond to the user. You can add code inside of it that will only run when something happens.
- Changing the second input changes the type of interaction that will make the code inside the `onEvent` run.
- Code outside the `onEvent` runs right away. Code inside an `onEvent` will only run when the event happens.
- Even if code is after an `onEvent`, it will run first if it's outside of any `onEvent`

Level 5: Run this level in the same way. Have students complete all three prompts and then bring the class together after a few minutes for a discussion.

Discuss: Have students share the results of their discussion and anything else they noticed. Here are good points to draw out.

- `playSound` will play a sound you pick from the Sound Library
- Lines that start with `//` are called comments and don't actually run. They just help you understand your code.

Level 6: Run this level in the same way. Have students complete all three prompts and then bring the class together after a few minutes for a discussion.

Discuss: Have students share the results of their discussion and anything else they noticed. Here are good points to draw out.

- Random number chooses a new random number each time, between the high and low value given
- `onEvent` takes many different event types, not just `"mouseOver"` and `"click"`. Depending on the situation different ones make more sense.

Wrap Up (15 mins)

Prompt: Think about your experiences today and in the previous lesson. How is a programming language different from natural language?

Remarks

> Awesome job! Today was our first chance to check out what programming in App Lab is like. So far we've only learned a few blocks but we've already seen they'll let us make a wide variety of types of programs. We're going to get a lot more time to practice using them, but before we do let's get some vocabulary in your journal to make sure we're using the same words to talk about what we saw today.

Journal: Go through each vocabulary word (Program Statement, Program, Sequential Programming, Event Driven Programming) and give students a chance to record each piece of information.

Remarks

> Great job today! You've learned a lot so far about programs. It's important to remember that programs need to work for a variety of inputs and

Discussion Goal

Goal: Make this a quick discussion to help connect the previous lesson to this one. Help bring out some of the following points.

- Programming languages are much more precise than natural language
- Programming languages have very strict rules
- Programming languages may feel a little awkward at first.

Teaching Tip

Reinforcing Vocabulary: A lot of the vocabulary introduced here is taken straight from the AP framework. The images are designed to help connect the definitions to the experiences they had in this lesson. Help students make these connections by not only writing down definitions but talking through how it's connected to what they saw.

> outputs. That's what makes programming interactive apps so fun! Today, you also learned how to describe the behavior of a program, or how the program works when it's run and how the user interacts with it.
>
> When we talk about how programs run, we can describe both what the program does and specifically how the program statements accomplish this goal.
>
> We're going to keep practicing using these words and going forward you're going to get more chance to practice programming. Start thinking about how you might want to use what you learned today in your project.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

- What is the difference between a sequential program and an event-driven program?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **AAP-3** - Programmers break down problems into smaller and more manageable pieces
- ▶ **CRD-2** - Developers create and innovate using an iterative design process

---

Lesson 7: Debugging

Overview

In this lesson students practice using the different programming concepts that they were introduced to in the last lesson. To begin, however, they are introduced to the concept of debugging and are encouraged to use and reflect on this practice throughout the lesson. At the end of the lesson students share their experiences debugging as well as an new realizations about programming.

Purpose

This lesson serves a number of roles. Even if students had modified programs in the previous lesson, this continues to be an introduction to many of the skills of programming. Students are also introduced to debugging as a skill they'll need to use and develop as programmers. A huge goal of this lesson, however, is attitudinal.

Agenda

Warm Up (5 mins)

- **Preview the Lesson**

Activity (30 mins)

- **Practice Time**

Wrap Up (10 mins)

- **Assessment: Check For Understanding**

View on Code Studio

Objectives

Students will be able to:

- Debug simple sequential and event-driven programs
- Use the debugging process and Identify specific best practices for debugging programs
- Use the speed slider, break points, and documentation as part of the debugging process

Preparation

- [ ] Review the steps of the debugging process
- [ ] Review the levels students will need to complete on Code Studio

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

Teaching Guide

Warm Up (5 mins)

Preview the Lesson

Prompt: Your friend calls and says "I can't get music to come out of my speakers, can you help?". Write a quick list of everything you'd ask them or have them check to try to fix the problem.

Remarks

> Today we're going to practice programming, but we're also going to practice a very important skill in programming called "debugging". Let's see what it looks like.

Discussion Goal

Optional Warmup Prompt: This prompt is optional and included for instances where you need a warm up prompt on the board when students walk in. If your classroom is able to move directly to main activity you should feel free to skip this prompt in the interest of time.

Goal: This prompt should help get students in the mindset of debugging. If students ask for more details tell them that this is all the information they have and that they should feel free to

Activity (30 mins)

Practice Time

Display: Show the Debugging video and then show the debugging process.

Remarks

> We're going to use this process to help us fix programs. I want you to use this process as you fix issues you find in code today. At the end of the lesson we'll talk more about this process and any specific strategies you document along the way.

Group: Place students in pairs.

Level 2: This level requires students to recognize that strings need to go in quotes.

Level 3: This level also requires students to recognize that strings need to go in quotes. Ideally students will get more familiar with checking for yellow warnings in this level.

Level 4: The code in this level will run, but the wrong text and sounds have been placed in the wrong places. This shows that code may have no warnings but still is not working correctly.

Level 5: Again this code has no warnings but in testing the app students should see that the range of values for the random numbers is too large. Iterative testing with different values should help them pick a number that keeps the smiley inside the screen.

Level 6: In this level an entire event handler is missing. Students will need to add it to the app.

Level 7: This level explicitly introduces the fact that event handlers (`onEvent`) should not go inside other `onEvent` blocks. This is explicitly addressed in the concluding slide show as well.

Level 8: This app is a good chance for students to practice adding functionality of their own. While nothing starts off as "broken", students will need to use debugging practices as they add code to this app.

Wrap Up (10 mins)

Prompt: If you were using the debugging process then you should have some notes of good debugging tips. Share those with your neighbor and add any new ones you forgot to add. Be ready to share with the class.

Display: Show the slide which lists some best practices in addition to those students may have mentioned.

Remarks

> Debugging is an important and entirely normal part of programming. You code won't always work the first time, and that's OK! Debugging is a skill that you can practice and get better at. Using documentation and leaving comments for yourself are important skills, but so is working with classmates or learning to more effectively search for bugs. We'll keep using these skills.

Journal: Have students add the following vocabulary words and definitions to their journals: documentation and comment.

Discussion Goal

Goal: This discussion should help reinforce the fact that debugging is a skill that can be learned and that it is made up of many little steps and understandings. Give the room an opportunity to share as many ideas as they can before sharing the provided slides with some recommended steps.

Remarks

> Let's take a minute to talk more about comments and documentation. Comments help explain the code, but do not affect how the program runs. They are meant to be read by people! When we write code, we don't only write for the computer we also write for other people. It's important that others can understand our code, so write your code clearly using the practices we discussed and comments.
>
> Not all programming environments support comments, so other forms of documentation may be important like keeping a separate document with information about your program. The key takeaway here, is no matter what the format, comments and documentation are important!
>
> As you grow in your programming skills, you will start to appreciate how valuable comments can be. You don't have to wait until a program is complete to write the comments. You should be doing this as you develop the project. There will be opportunities to write comments called out in the App project you are currently working on.

Assessment: Check For Understanding

Check For Understanding Question(s) and solutions can be found in each lesson on Code Studio. These questions can be used for an exit ticket.

- What is one thing you really enjoyed about today's activity?
- Is there anything that you found confusing or need extra help with?

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **CS** - Computing Systems

CSP2021

- ▶ **AAP-2** - The way statements are sequenced and combined in a program determines the computed result
- ▶ **CRD-2** - Developers create and innovate using an iterative design process

Lesson 8: Project - Designing an App Part 3

Overview

Student learn about Pair Programming by watching a video and then practicing it themselves while working on their project apps. At this stage, students are adding their first lines of code to their app using debugging skills from the previous lesson.

Purpose

Pair Programming is an effective collaboration strategy both inside the classroom and in professional setting. As this is the first opportunity students have to program starting from a blank screen, this is a good opportunity to explore the usefulness of Pair Programming.

Agenda

Warm Up (5 mins)
- **What Makes a Good Partner?**

Activity (35 mins)
- **Using Pair Programming**

Wrap Up (5 mins)
- **Reflecting on Pair Programming**

View on Code Studio

Objectives

Students will be able to:

- Effectively use pair programming while designing the features of an app
- Create the code and user interface of an app based on a program specification

Preparation

- [ ] Make sure students have access to their App Development Planning Guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

What Makes a Good Partner?

Prompt: What makes a good partner?

Remarks

> Now you've got the skills you'll need for your project, but we'll need a minute to talk about how to use them when working in teams. Today you're mostly going to have work time, but you're going to be practicing a new skill called pair programming.

Discussion Goal

Goal: Make this a quick discussion and just aim to get a few ideas from around the room. For example, a good partner...

- listens
- contributes
- shares the work evenly
- and so on....

Activity (35 mins)

Using Pair Programming

> **Group:** Place students in pairs with their project partner
>
> **Distribute:** Direct students back to the **App Development Planning Guide - Activity Guide**
>
> **Step 5:** Students fill out the chart on page 4, listing all of the Event Handlers in their programs. They should be able to determine these based on the Program Specification they designed.

Display: Play the Pair Programming video found either in **CSP Unit 3 - Intro to App Design - Presentation** or on Code Studio. Afterwards review the pair programming steps found in Code Studio.

Remarks

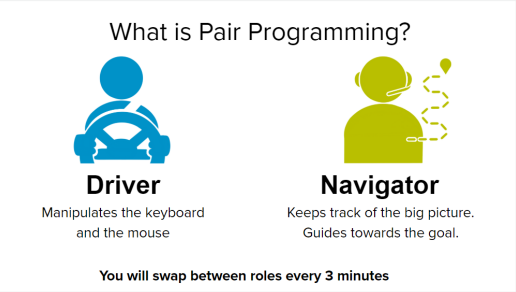
> Today we're going to use the pair programming as you work on your app with your partner. To begin we'll make the partner sitting on the left the driver and the one on the right the navigator. Every few minutes I'll ask you to switch roles.

Circulate: Give students time to work on their project. As they do so circulate the room encouraging them to use debugging practices they've learned in previous lessons. Every few minutes ask left and right partners to switch being drivers and navigators.

Teaching Tip

Taking on the Blank Screen: This is the first time students are adding code to a blank screen. Some students may need assistance. Direct them back to their Planning Guides to help students understand the `onEvent` blocks they need to add to their project.

Supporting Pair Programming: Your biggest role in supporting pair programming will be encouraging students to use it early on and helping enforce the switching. Run the timer in the slides and have left and right partners switch roles. You may opt to increase the time to longer periods as time goes on.



Debugging Practices: Students learned a lot of debugging practices in the previous lesson. Encourage students to use these and record bugs they find as they program today. Continue to normalize and celebrate debugging as a normal and fun part of programming.

Wrap Up (5 mins)

Reflecting on Pair Programming

Prompt: How does Pair Programming help when working on a project? How does it help with the debugging process in particular?

Journal: Have students add "Pair Programming" to their journals.

Discussion Goal

Goal: Use this prompt to reinforce not only the value of pair programming and collaboration while programming, but also to remind students of debugging practices they learned in the previous lesson. Possible answers include

- Talking through my code with another person helps me figure out what to do
- When I get stuck, there's someone there to help brainstorm a solution
- There's more than one set of eyes to find bugs
- Different partners bring different perspectives to a project

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

CSP2021

▶ **CRD-1** - Incorporating multiple perspectives

Lesson 9: Project - Designing an App Part 4

Overview

Students continue working on their apps. Halfway through class the focus of the lesson shifts to getting feedback. Students watch other groups test their apps and collect feedback that will be used to make updates.

Purpose

This lesson reinforces the importance of feedback in the process of developing software. Feedback and testing help make sure that an app actually works for lots of people and in the context in which it will be used. This process also reinforces the importance of making iterative improvements in making software.

Agenda

Warm Up (5 mins)
Activity (30 mins)
- **Work Time - 25 mins**
- **Testing and Feedback - 10 mins**

Wrap Up (10 mins)

View on Code Studio

Objectives

Students will be able to:

- Test an app's functionality by attempting to use features and behavior described in a program specification
- Provide effective feedback on the functionality or usability of an app
- Iteratively improve an app based on feedback

Preparation

- [ ] Make sure students have access to their App Development Planning Guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

Prompt: Think of times when you've received helpful feedback on school work, a hobby, or a sport.

- What makes good feedback?
- What makes bad feedback?

Remarks

> Today we're going to keep working on our apps, but we're also going to be getting feedback from our classmates. Let's get to it!

Discussion Goal

Goal: This is an optional prompt if your class situation requires it. If you believe your students need more time to work on their apps then move right into work time. This prompt, however is designed to get students thinking early about the differences between useful and unhelpful feedback. For example, students may note that...

- Good feedback:
  - Specific information beyond "it's great".
  - Explains why something needs work
- Bad Feedback:
  - Is overly negative without constructive feedback
  - Does not go into enough detail

Activity (30 mins)

Group: Place students back in pairs with their project partners.

Work Time - 25 mins

Remarks

> For the next twenty five minutes, you and your partner will use Pair Programming to continue working on your app. After this is done, you will get feedback from another group.

Circulate: Direct students to Code Studio to continue working on their apps.

Teaching Tip

Review Pair Programming: You may want to reuse the slides from the previous lesson to remind students of their Pair Programming roles.

Encourage Debugging Practices: Continue to encourage students to use the debugging practices they learned in previous lessons and normalize and celebrate debugging as a part of programming

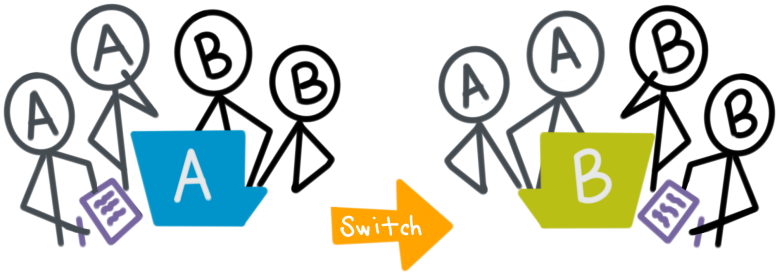
Testing and Feedback - 10 mins

Distribute: Direct students back to the **App Development Planning Guide - Activity Guide**.

Step 6: Groups swap apps and gather feedback. It is ok to move ahead to this section even if students are not done with their apps yet.

Here's what you should observe:

- Group A lets Group B test their app.
- Group A watches Group B use the app, and writes down improvements that come to mind.
- Group A interviews Group B asking for specific improvements Group A can make. This may be different than what Group A came up with during the observation. These improvements are written down in the Activity Guide.
- Group A and B swap and now Group A tests Group B's app while Group B watches.
- After this, mix up the group pairings and repeat the steps above.



Wrap Up (10 mins)

Remarks

> **Step 7:** Now return to your stations and pick at least one improvement to make to their app based on feedback. Write this down in the Planning Guide in Step 7.

Prompt: Why is it important to get feedback from others while building your app? What is the value of getting this feedback even if you aren't finished with your app?

Remarks

> Feedback and testing are important parts of making good software. They ensure that our apps actually work the way we designed them, and they help us make gradualy improvements. Tomorrow you'll have a chance to make some final touches on your app before we submit them!

Discussion Goal

Goal: Use this prompt as time allows. The goal is to reinforce the importance of feedback in the development of good software. Possible answers include:

- Another person may find a bug that you have been overlooking
- Something that seems obvious to you is not obvious to the user
- Even if the app isn't finished, feedback can help guide the next steps

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- **AP** - Algorithms & Programming

CSP2021

- **CRD-1** - Incorporating multiple perspectives
- **CRD-2** - Developers create and innovate using an iterative design process

Lesson 10: Project - Designing an App Part 5

Overview

Students complete their apps, making any final adjustments based on feedback from their peers. Students spend some time reviewing other apps that classmates made and then complete a short set of reflection prompts before submitting their projects.

Purpose

This lesson concludes the project students have been working on throughout the unit. Students will have a chance to respond to any feedback they received and incorporate that into their project. The reflection prompts in the activity help students synthesize their overall experience with a focus on how collaboration within their team and within their classroom impacted the final projects they designed.

Agenda

Warm Up (5 mins)
Activity (30 mins)
- **Finish building apps**

Wrap Up (10 mins)
- **Assessment: Project**

View on Code Studio

Objectives

Students will be able to:

- Reflect on the value of different stages of a development process in creating an app

Preparation

- [ ] Make sure students have access to their App Development Planning Guides
- [ ] Review the two sample project submissions, available in the "For Teachers Only" area on the first level of this lesson on Code Studio

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

Remarks

> Your apps are almost done and soon will be ready to share with the world. Let's get back to work.

Teaching Tip

Get to the Project: Quickly move the class to the activity where the bulk of their time should be spent today.

Activity (30 mins)

Group: Place students back in pairs with their project partner.

Distribute: Make sure students have access to the **App Development Planning Guide - Activity Guide**

Finish building apps

Remarks

> For the next thirty minutes, you and your partner will work on finishing your app. Don't forget to consider the suggestions your classmates made. Check the rubric before submitting your app.

Code Studio: Direct students to continue working on their app. When they are finished, they should hit "Submit" to turn in their app.

Wrap Up (10 mins)

Reflection: Direct students to the Reflection Section in the Planning Guide. There are two questions for students to complete.

Submit Projects: Students should submit their final projects including both a link to the app and their planning guide.

Assessment: Project

Use the rubric to assess student projects. The rubric can be found on the last page of the **App Development Planning Guide - Activity Guide**.

Standards Alignment

CSTA K-12 Computer Science Standards (2017)

- ▶ **AP** - Algorithms & Programming

Lesson 11: Assessment Day

Overview

Students complete the Unit Assessment and then spend the rest of class sharing their projects.

Purpose

This lesson provides a bookend to Unit 3 where students demonstrate their learning through their projects and a multiple choice assessment.

Agenda

Warm Up (5 mins)
Activity (30 mins)
- **Unit Assessment - 25 mins**
- **Gallery Walk - 10 mins**

Wrap Up (10 mins)

View on Code Studio

Objectives

Students will be able to:

- Reflect on the value of different stages of a development process in creating an app

Preparation

- [ ] Make sure students have access to their App Development Planning Guides

Links

Heads Up! Please make a copy of any documents you plan to share with students.

For the Teachers

- **CSP Unit 3 - Intro to App Design** - Presentation

For the Students

- **App Development Planning Guide** - Activity Guide Make a Copy

Teaching Guide

Warm Up (5 mins)

Remarks

> We've come to the end of the unit, and we've learned a lot about developing apps. Today we are going to take an assessment and then you will have the opportunity to share your apps with the entire class.

Activity (30 mins)

Unit Assessment - 25 mins

Do This: Students complete the Unit Assessment on Code Studio.

Gallery Walk - 10 mins

Do This: One student per group displays their app on the computer. Groups rotate around the room running the different apps. Students can leave comments on sticky notes at each station.

Remarks

> If you want, your app can be shared with people beyond this classroom. Click the "Share" button and then grab the link or type in a phone number to share it with a friend or family member. Congratulations on making your first app!

Wrap Up (10 mins)

Do This: If you have time, quickly review the Assessment.