Lesson 25: Functions

45 minutes

Overview

In previous lessons, students have learned to use a number of abstractions in their programs which have allowed them to build much more complex programs while ignoring the details of how that behavior is created. In this lesson, students learn to build abstractions of their own by creating functions that will serve to organize their code, make it more readable, and remove repeated blocks of code. Students first think about what sorts of new blocks they would like in Game Lab, and what code those blocks would contain inside. Afterward, students learn to create functions in Game Lab. They will use functions to remove long blocks of code from their draw loop and to replace repeated pieces of code with a single function.

Question of the Day: How can programmers use functions to create their own abstractions?

Standards

Full Course Alignment

CSTA K-12 Computer Science Standards (2017)

▶ **AP** - Algorithms & Programming

Agenda

Objectives

Students will be able to:

- Create and use functions for blocks of code that perform a single high-level task within a program
- Explain how functions allow programmers to reason about a program at a higher level
- Explain the advantages of using functions in a program

Preparation

- Check the **"Teacher's Lounge"** forum for verified teachers to find additional strategies or resources shared by fellow teachers
- If you are teaching virtually, consider checking our **Virtual Lesson Modifications**

Links

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

For the teachers

- **Functions** - Slides ▾ Make a Copy

For the students

- **Functions** - Video (**Download**)
- **Functions** - Resource

Vocabulary

- **Function** - A named bit of programming instructions.

Introduced Code

- `function myFunction() { // function body, including optional '`
- `myFunction();`

Teaching Guide

Warm Up (5 minutes)

Your Personal Blocks

Prompt: What's one block you'd like to have in Game Lab? What would it do? What code would it use to work?

Think-Pair-Share: Allow students to explain their blocks to a partner before sharing out their ideas.

Discussion Goal: As students share their ideas, make sure they have a clear idea of the code that they would use to make the block, reminding them that all of the blocks they have learned this chapter (e.g. velocity, collisions) have been built from blocks that they already knew.

Remarks

> Today, we're going to learn how to create our own blocks so that we decide exactly how they will work. These special blocks are called functions, and they are one of the most powerful parts of programming.

Question of the Day: How can programmers use functions to create their own abstractions?

Activity (35 minutes)

Video: Show students the **Functions** video in the slides.

Teaching Tip

To encourage active engagement and reflection, use one or more of the strategies discussed in the **Guide to Curriculum Videos**.

Questions to Consider with Video:

- Think of a time when a function might have helped you write a program.
- What code would go in the **definition** of the function?
- When would you **call** the function?
- What would you name it?

Discussion Goal: Ensure students know the key vocabulary term **Function** and its definition as *a named bit of programming instructions.* Make sure students understand the role of the two steps in using functions, as well as seeing functions as a form of "chunking" or abstraction. The function definition should include all of the code that they want to run, and the name of the function should be a short description of the purpose of the code. The function should be called at each place in the program where the student wants that block of code to run.

Transition: Move students into Code Studio where they will learn to create and call functions.

The first prediction level program is the first time they will see functions in action.

Discuss: Students should consider and discuss:

- What will be drawn on the screen and why?

Discussion Goal: Predictions will vary but after viewing the Functions video, students should be able to predict that the program will draw what is inside the `drawBackground` and `drawPlanet` blocks of code. It is important to point out that the `drawPlanet` function was called twice which is why there are two planets drawn on the screen, each one a different color and location. Students should also notice the `drawStar` function being called *within* the `drawBackground` function multiple times rather than at the top of the program like the other function calls. You can take this opportunity to discuss how this helped organize and simplify the code since the stars are part of the background.

1 **Predict**

Teaching Tip

Guide to Programming Levels: Additional guidance for programming levels is provided in the **CSD Guide to Programming Levels**. This document includes strategies and best-practices for facilitating programming levels with students.

2-3 **Skill Building**

2 3

Teaching Tip

Introducing Functions

In these first few levels students are just being shown the syntax of functions and are not asked to write or create their own. It can be useful to explain creating a function as basically "creating a new block" just like another programmer created the "isTouching" or "velocity" blocks that they've seen actually contain other more complex code.

4 **Predict**

5 Practice

Teaching Tip

Why Use Functions

Level 4a, 4b, and 4c introduce three uses of functions, namely removing repetition in programs, allowing code to quickly be changed at multiple points, and providing organization in code. Students will need to write more of their own functions in these levels.

6 Quick Check

7-9 Collector Game

7 8 9

Teaching Tip

Level 6: Functions in Context

In these levels, students will use functions to organize code within a simple game. While it is not identical to the side scroller, many of the skills and uses of functions in these levels can and should be used when they complete their side scroller.

Assessment Opportunity

Formative Assessment: This level can be used as a formative assessment. A rubric is provided in the level, and written feedback can be given to students. **Click here to learn more about giving feedback to students**.

10 Challenges

Wrap Up (5 minutes)

Key Vocabulary:

- **Function** - a named bit of programming instructions

Prompt: Why would we say that functions allow us to "create our own blocks?" Why is this something we'd want to do? Why would a function count as an abstraction?

Discuss: Have students discuss at their table before talking as a class.

Assessment Opportunity

Goal: Use this first prompt to review what students learned today. When they create a function they are creating their own block that they can call or use whenever they like. They saw at least two primary motivations for creating functions today including:

- Simplifying code by breaking it into logically named chunks
- Allowing a programmer to think at a higher level by hiding the details or a particular bit of programming instructions
- Avoiding repeated code by making one block you can use multiple times

Students should review the definition of abstraction as "a simple way of representing something complex". Note that a function is an abstraction because it allows you to create one simple name for a more complex block of code.

Remarks

> Functions are a useful tool for helping us write and organize more complex pieces of code. As we start looking to the end of the unit and your final project, being able to keep your code organized will be an important skill.

Question of the Day: How can programmers use functions to create their own abstractions?