CodeHS

Student Learning Outcomes for Introduction to Computer Science

CodeHS is a self-paced introductory computer science curriculum inspired by Stanford's introductory computer science class. Through a series of learning modules, including video tutorials, quizzes, example code, applied programming exercises, and programming challenges, the CodeHS curriculum teaches the foundations of computer science and basic programming. Once students complete the CodeHS Introduction to Computer Science course, they will have learned material equivalent to a semester college introductory course in Computer Science and be able to program in JavaScript.

No Prerequisites:

There are no prerequisites for this course. Students don't even have to know what it means to code! As long as your school has computers and internet access, your students are ready to learn. CodeHS' mission is to make computer science fun and accessible to absolute beginners, so anyone interested in learning to code can sign up for a CodeHS membership and start learning.

Accessible Content:

Content on CodeHS is broken down into modules. Each module teaches a set of concepts that are fundamental to the study of computer science. Students watch short video tutorials and experiment with example code to learn about each concept. They then complete short programming exercises to demonstrate their understanding of the material.

Practical Outcomes:

CodeHS is taught in JavaScript. Once students complete the first 3 modules on CodeHS, they will have made a game. After the 4th module, they'll be equipped to make anything on our demos page on the site. After completing the CodeHS introductory Computer Science curriculum, students will know JavaScript and have the foundation in the basics of programming.

Here is a breakdown of the important programming skills that students will develop concepts that they will learn about by working through the CodeHS curriculum:

Programming with Karel: Teaches what it means to "program" and allows students to focus on solving problems using code, rather than getting bogged down in syntax. Students solve problems by moving Karel the Dog around the grid.

<u>Students will know:</u>
-the definition of a command
-the definition of code, computer program and the relationship between the two
-the definition and purpose of a function
-Basic syntax rules for commands, loops, and functions
-How to use a loop to repeat code
-Precondition/postcondition.
-Commenting code to make it clearer for others
-The differences amongst for loops, while loops, and if/else statements.
-Appropriate control structures that help to meet an objective.
-The purposes and benefits using functions.
-Top down design and problem decomposition

<u>Students will be able to:</u>
-Write a sequence of commands to solve a simple puzzle
-Complete sequential programming
-Write a function that eliminates repeated code
-Correctly use a loop to repeat code
-Execute a code that repeats an appropriate number of times.
-Choose the appropriate control structure from amongst if, if/else, and while statements
-Detect and address bugs (errors) in a program
-Apply best practices in writing code
-Use top-down design and decomposition to solve a multi-part problem

Basic Javascript and Graphics: Introduces the basics of JavaScript, including variables, user input, control structures, functions with parameters and return values, and basic graphics, how to send messages to objects.

<u>Students will know:</u>
-variables are used to store a value
-user input allows for our programs to be interactive
-booleans are true/false values
-logical operators connect and modify boolean expressions
-comparison operators compare values
-randomization of numbers, colors, and graphics
-how to write more general functions in JavaScript
-return values and parameters are used to create reusable code
-how scoping works for local variables

Students will be able to:
-declare, initialize and change the values of variables
-perform mathematical operations using Javascript
-abstract and modularize a program using functions and parameters
-draw graphics and shapes
-create comparison statements using Boolean values
-increment a value
-write a function that takes parameters and can return values.
-write a reusable function
-write a more general for loop
-use variables within a for loop

Animation and Games: Watch graphics come to life! Teaches how to make objects move around the screen and how to let the user interact with programs using the mouse. At the end of this section, students will program their own video game.

Students will know:
-how programs can be user-driven and interactive through the use of timers, mouse events and keyboard events

Students will be able to:
-complete a program that uses timers, randomization, and mouse clicks in combination.
-apply computational concepts and produce an interactive program utilizing graphics and event-driven game.

Basic Data Structures: Introduces lists/arrays, maps/objects, sets, and grids. These are the essential basic data structures that any program will use.

Students will know:
-What is an array?
-An array is an ordered list of elements
-What is a set?
-A set stores unordered elements
-What is a object?
-Objects or maps have keys mapping to values
-What is a grid?
-How to loop through arrays.
-Finding and removing an element in a list.
-the function and purpose of objects.
-the function and purpose of maps.
-the function and purpose, of grids.
-the relationship amongst arrays, lists, objects/maps, and grids.

Students will be able to:

-organize data using various types of data sets.
-manipulate data in an array, list, object, map, and grid.
-create objects, lists, sets, and grids
-synthesize the relationship amongst arrays, lists, objects, maps and grids.
-iterate through data sets and manipulate individual elements.

Game Design Components: Walk through the creation of the classic Helicopter game one step at a time.

Students will know:
-what elements make a computer program a game.
-the process of creating side to side movement.
-the process of designing acceleration and deceleration of a graphic object.
-the process of designing the appearance of gravity in a game.
-the process of creating platform movement.
-the process of compound data to create objects.

Students will be able to:
-extend their knowledge of graphics, basic computational concepts, data manipulation, and user interaction to produce an interactive and dynamic helicopter game.