Greenwich Public Schools Curriculum Overview

INTRODUCTION TO COMPUTER PROGRAMMING A

Personalized learning is achieved through standards-based, rigorous and relevant curriculum that is aligned to digital tools and resources.

Note: Teachers retain professional discretion in how the learning is presented based on the needs and interests of their students.

Course Description

Introduction to Computer Programming A
(Concentration in JavaScript)
1st or 2nd semester
027205/027206 6 Blocks .5 Credit
Prerequisites: C or better in Algebra 1, Extended Algebra, or Algebra/Geometry Course 2.
This semester course is designed to introduce the most fundamental web-based concepts and how to use them in JavaScript. Topics include data types, functions, loops, control flow, canvas, and interactivity. Emphasis is placed on real-world understanding through a project-based learning environment. This course is the foundation for the next sequence of computer science classes.

Unit Guide

Unit 0: Introduction to Programming
Unit 1: Introduction to JavaScript
Unit 2: Binary and Hexadecimal
Unit 3: Conditionals
Unit 4: Arrays
Unit 5: The DOM
Unit 6: Final Project

Computational Thinking Practices

- Program Design and Algorithm Development
- Code Logic
- Code Implementation
- Code Testing
- Documentation

Mathematical Practices

- Make sense of problems and persevere in solving them.
- Reason abstractly and quantitatively.
- Use appropriate tools strategically.
- Attend to precision.
- Look for and make use of structure.
- Look for and express regularity in repeated reasoning.

Enduring Understandings

- Code 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.
- Programs can be used to process data, which allows users to discover new information and create new knowledge.
- The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- 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.

Essential Questions

- What types of problems can be solved more easily with a computer, and what types can be solved more easily without a computer? Why?
- How does one use an IDE and event-driven programming to build an app?
- How do multiple programming languages work together in a computer application? What are the roles of HTML, CSS and javaScript in a browser-based application? What are the specific syntax rules of each language?
- How can we store data in a program to solve problems?
- What is the binary number system that underlies all digital representation? How can binary numbers be used to represent all digital data?
- How do video games group the different actions for a player based on what key is pressed on the keyboard or controller? How do apps group different actions together based on user interaction, such as pressing buttons?
- How can students use if or else statements to control programs? How can functions be called from various locations in a program? What are the advantages of nesting a conditional statement within another conditional statement?
- How do variables of both simple and structured data, such as, lists, enable us to manage the complexity of a program?
- What might happen if you completed the steps in your regular morning routine to get ready and go to school in a different order? How might the reordering affect the decisions you make each morning?
- Why is there a need to debug?
- Why is it important to test code on a variety of test cases?
- How does testing the program with a wide range of values help confirm its effectiveness?

Resources and Assured Experiences

Online resources:

W3schools.com
Khan Academy
Code.org

Caret IDE, repl.it once it completes digital tool authorization

Summative Assessments
Formative Assessments
Classwork
Preparation

GHS Capstone Task: Capstone Vision of the Graduate Capacity: #6 - *Generates innovative, creative ideas and products*

Quarterly Grading - Quarter Grades will be determined using the following components:

- Participation (includes Classwork) = 20%
- Preparation (includes Homework) = 20%
- Assessments (both Summative & Formative) = 60%

CSTA Computer Standards:

3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

3A-AP-14 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.

3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

A-AP-18 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

3A-AP-19 Systematically design and develop programs for broad audiences by incorporating feedback from users.

3B-AP-10 Use and adapt classic algorithms to solve computational problems.

3B-AP-14 Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

3B-AP-22 Modify an existing program to add additional functionality and discuss intended and unintended implications (e.g., breaking other functionality).