



ESL
SCIENCE
BUSINESS
BILINGUAL
PRESCHOOL
MATHEMATICS
LIBRARY MEDIA
SOCIAL STUDIES
WORLD LANGUAGES
GIFTED & TALENTED
TECHNOLOGY EDUCATION
ENGLISH LANGUAGE ARTS
FINE & PERFORMING ARTS
FAMILY & CONSUMER SCIENCE
HEALTH & PHYSICAL EDUCATION

RAHWAY PUBLIC SCHOOLS

CURRICULUM & INSTRUCTION

Content Area: Mathematics

Course: Exploring Computer
Science

Grade Level: 9-12

This curriculum is part of the Educational Program of Studies of the Rahway Public Schools.

ACKNOWLEDGMENTS

Jeffery Kurczeski,

Program Supervisor of 7-12 Math & Science and 9-12 Business & Technology Education

The Board acknowledges the following who contributed to the preparation of this curriculum.

Ambika Bhosale, Computer Science Teacher

Dr. Tiffany A. Beer, Director of Curriculum and Instruction

Subject/Course Title:
Exploring Computer Science
Grades 9-12

Date of Board Adoption:
August 26, 2025

RAHWAY PUBLIC SCHOOLS CURRICULUM

Exploring Computer Science: Grades 9-12

PACING GUIDE

Unit	Title	Pacing
1	Karel the Dog	8 weeks
2	JavaScript and Graphics	9 weeks
3	JavaScript Control Structures	9 weeks
4	Functions and Parameters	6 weeks
5	Animation and Games	8 weeks

ACCOMMODATIONS

<p>504 Accommodations:</p> <ul style="list-style-type: none"> ● Provide scaffolded vocabulary and vocabulary lists. ● Provide extra visual and verbal cues and prompts. ● Provide adapted/alternate/excerpted versions of the text and/or modified supplementary materials. ● Provide links to audio files and utilize video clips. ● Provide graphic organizers and/or checklists. ● Provide modified rubrics. ● Provide a copy of teaching notes, especially any key terms, in advance. ● Allow additional time to complete assignments and/or assessments. ● Provide shorter writing assignments. ● Provide sentence starters. ● Utilize small group instruction. ● Utilize Think-Pair-Share structure. ● Check for understanding frequently. ● Have student restate information. ● Support auditory presentations with visuals. ● Weekly home-school communication tools (notebook, daily log, phone calls or email messages). ● Provide study sheets and teacher outlines prior to assessments. ● Quiet corner or room to calm down and relax when anxious. ● Reduction of distractions. ● Permit answers to be dictated. ● Hands-on activities. ● Use of manipulatives. ● Assign preferential seating. ● No penalty for spelling errors or sloppy handwriting. ● Follow a routine/schedule. ● Provide student with rest breaks. ● Use verbal and visual cues regarding directions and staying on task. ● Assist in maintaining agenda book. 	<p>IEP Accommodations:</p> <ul style="list-style-type: none"> ● Provide scaffolded vocabulary and vocabulary lists. ● Differentiate reading levels of texts (e.g., Newsela). ● Provide adapted/alternate/excerpted versions of the text and/or modified supplementary materials. ● Provide extra visual and verbal cues and prompts. ● Provide links to audio files and utilize video clips. ● Provide graphic organizers and/or checklists. ● Provide modified rubrics. ● Provide a copy of teaching notes, especially any key terms, in advance. ● Provide students with additional information to supplement notes. ● Modify questioning techniques and provide a reduced number of questions or items on tests. ● Allow additional time to complete assignments and/or assessments. ● Provide shorter writing assignments. ● Provide sentence starters. ● Utilize small group instruction. ● Utilize Think-Pair-Share structure. ● Check for understanding frequently. ● Have student restate information. ● Support auditory presentations with visuals. ● Provide study sheets and teacher outlines prior to assessments. ● Use of manipulatives. ● Have students work with partners or in groups for reading, presentations, assignments, and analyses. ● Assign appropriate roles in collaborative work. ● Assign preferential seating. ● Follow a routine/schedule.
<p>Gifted and Talented Accommodations:</p> <ul style="list-style-type: none"> ● Differentiate reading levels of texts (e.g., Newsela). ● Offer students additional texts with higher lexile levels. ● Provide more challenging and/or more supplemental readings and/or activities to deepen understanding. ● Allow for independent reading, research, and projects. ● Accelerate or compact the curriculum. ● Offer higher-level thinking questions for deeper analysis. ● Offer more rigorous materials/tasks/prompts. ● Increase number and complexity of sources. ● Assign group research and presentations to teach the class. ● Assign/allow for leadership roles during collaborative work and in other learning activities. 	<p>ELL Accommodations:</p> <ul style="list-style-type: none"> ● Provide extended time. ● Assign preferential seating. ● Assign peer buddy who the student can work with. ● Check for understanding frequently. ● Provide language feedback often (such as grammar errors, tenses, subject-verb agreements, etc...). ● Have student repeat directions. ● Make vocabulary words available during classwork and exams. ● Use study guides/checklists to organize information. ● Repeat directions. ● Increase one-on-one conferencing. ● Allow student to listen to an audio version of the text. ● Give directions in small, distinct steps. ● Allow copying from paper/book. ● Give student a copy of the class notes.

- Provide written and oral instructions.
- Differentiate reading levels of texts (e.g., Newsela).
- Shorten assignments.
- Read directions aloud to student.
- Give oral clues or prompts.
- Record or type assignments.
- Adapt worksheets/packets.
- Create alternate assignments.
- Have student enter written assignments in criterion, where they can use the planning maps to help get them started and receive feedback after it is submitted.
- Allow student to resubmit assignments.
- Use small group instruction.
- Simplify language.
- Provide scaffolded vocabulary and vocabulary lists.
- Demonstrate concepts possibly through the use of visuals.
- Use manipulatives.
- Emphasize critical information by highlighting it for the student.
- Use graphic organizers.
- Pre-teach or pre-view vocabulary.
- Provide student with a list of prompts or sentence starters that they can use when completing a written assignment.
- Provide audio versions of the textbooks.
- Highlight textbooks/study guides.
- Use supplementary materials.
- Give assistance in note taking
- Use adapted/modified textbooks.
- Allow use of computer/word processor.
- Allow student to answer orally, give extended time (time-and-a-half).
- Allow tests to be given in a separate location (with the ESL teacher).
- Allow additional time to complete assignments and/or assessments.
- Read question to student to clarify.
- Provide a definition or synonym for words on a test that do not impact the validity of the exam.
- Modify the format of assessments.
- Shorten test length or require only selected test items.
- Create alternative assessments.
- On an exam other than a spelling test, don't take points off for spelling errors.

UNIT 1 OVERVIEW

Content Area: Computer Science

Unit Title: Karel the Dog

Target Course/Grade Level: Exploring Computer Science/Grades 9-12

Unit Summary: In this unit, students are introduced to computer programming through Karel the Dog. Karel only knows how to move, turn left, put down, and pick up tennis balls in a grid world. Students will learn to give Karel these commands to instruct him to do certain things. Various coding concepts like functions, conditional statements, and iterative loops will be implemented using visual representations of Karel the dog. In doing so, students will discover what it means to program and enhance their problem-solving skills.

Approximate Length of Unit: 8 weeks

LEARNING TARGETS

NJ Student Learning Standards:

- 8.1.12.AP.1** Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.3** Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
- 8.1.12.AP.5** Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 8.1.12.AP.6** Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 8.2.12.ED.1** Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.

Career Readiness, Life Literacies, and Key Skills:

- 9.3.IT-PRG.4** Demonstrates the effective use of software development tools to develop software applications.
- 9.3.IT-PRG.5** Apply an appropriate software development process to design a software application.
- 9.3.IT-PRG.6** Program a computer application using the appropriate programming language.
- 9.4.12.CI.1** Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
- 9.4.12.CT.1** Identify problem-solving strategies used in the development of an innovative product or practice.

Interdisciplinary Connections and Standards:

ELA

- RI.CI.11–12.2** Determine two or more central ideas of an informational text and analyze how they are developed and refined over the course of a text, including how they interact and build on one another to provide a complex account or analysis; provide an objective summary of the text.

RI.MF.11–12.6 Synthesize complex information across multiple sources and formats to develop ideas, resolve conflicting information, or develop an interpretation that goes beyond explicit text information (e.g., express a personal point of view, new interpretation of the concept).

SL.II.11–12.2 Integrate multiple sources of information presented in diverse formats and media (e.g., visually, quantitatively, orally) in order to make informed decisions and solve problems, evaluating the credibility and accuracy of each source and noting any discrepancies among the data.

Mathematics

A.SSE.A.1 Interpret expressions that represent a quantity in terms of its context. a. Interpret parts of an expression, such as terms, factors, and coefficients. b. Interpret complicated expressions by viewing one or more of their parts as a single entity.

A.SSE.A.2 Use the structure of an expression to identify ways to rewrite it.

A.SSE.B.3 Choose and produce an equivalent form of an expression to reveal and explain properties of the quantity represented by the expression.

a. Factor a quadratic expression to reveal the zeros of the function it defines.

b. Complete the square in a quadratic expression to reveal the max or min value of the function it defines.

c. Use the properties of exponents to transform expressions for exponential functions.

Unit Understandings:

Students will understand that...

- Computer programming is a sequence of commands that are read by a computer in a language it can understand.
- Commands are functions that can be created by the user and then used by the user as they see fit.
- Control structures help programmers create efficient, organized programs.

Unit Essential Questions:

- What is programming, and how does it work?
- What is a computer command?
- How can computers “get stuck” while trying to execute commands?
- What is a control structure?
- How are complex problems broken down into smaller problems through programming?

Knowledge and Skills:

Students will know...

- The four basic Karel commands: move, turnLeft, putBall, and takeBall.
- The difference between defining a function versus calling a function.
- How to create a function.
- Where the “beginning” of a program is.
- What an if/if-else statement does.
- The difference between a for loop and a while loop.
- How to comment code.
- The importance of “top-down” design when it comes to programming.
- The purpose of control flow is to guide program execution.
- That functions promote code reusability and organization.
- How decomposition helps break down large problems into smaller, manageable sub-problems.
- The syntax and structure of if, if-else statements, for loops, and while loops.
- The conditions under which each type of loop (for vs. while) is most appropriate.
- The importance of debugging and common strategies for identifying and fixing errors in code.
- That program efficiency and readability are important considerations in software design.
- How to interpret and trace simple Karel programs.

Students will be able to...

- Complete various programming tasks on CodeHS.
- Use functions, loops, and conditionals to solve complex programming tasks.
- Explain how their code works using comments.
- Decompose complex Karel tasks into smaller, more manageable functions.
- Write custom functions to encapsulate repeated sequences of Karel commands.
- Apply conditional statements (if, if-else) to make Karel's actions dependent on specific conditions in the grid world (e.g., frontIsClear(), ballsPresent()).
- Implement iterative loops (for and while) to repeat actions a fixed number of times or until a certain condition is met.
- Combine functions, conditionals, and loops to solve more intricate programming challenges.
- Debug their Karel programs by identifying logical errors and syntax mistakes.
- Refactor their code to improve readability, efficiency, and adherence to programming best practices.
- Collaborate effectively on programming tasks using paired-programming techniques.
- Articulate their problem-solving process and explain the logic behind their Karel programs.

EVIDENCE OF LEARNING

Assessment:

What evidence will be collected and deemed acceptable to show that students truly “understand”?

- End of Unit Common Assessment - See folder for assessment links.
 - Students will take a test for the unit in which they will have to identify terminology, as well as the key concepts of the unit. Students will also be given a program to solve using programming and explain their thought process for the solution code.
- CodeHS.com lesson exercises
- Class participation
- Class discussion

Learning Activities:

What differentiated learning experiences and instruction will enable all students to achieve the desired results?

- Group programming projects
- Teacher demonstrations over the Screen Share software
- Reinforcement worksheets and extra practice
- Paired-programming challenges
- Individualized student-to-teacher code demonstrations

RESOURCES

Teacher Resources:

- Demonstrations of worked-out solutions from CodeHS.com
- Teacher designed worksheets
- CodeHS.com lesson exercises and videos

Equipment Needed:

- Chromebooks
- Access to high-speed internet
- CodeHS.com login

UNIT 2 OVERVIEW

Content Area: Computer Science

Unit Title: JavaScript and Graphics

Target Course/Grade Level: Exploring Computer Science/Grades 9-12

Unit Summary: In this unit, students will learn the basics of the real programming language called JavaScript. Students will apply many concepts and ideas learned from programming with Karel to help them create their first JavaScript programs, while at the same time, are learning about some distinct differences. Key new concepts, such as variables and user input, provide students with an entirely new dynamic when designing and solving programming tasks. Students will also be introduced to the graphics functions and how to manipulate various properties of each graphical object.

Approximate Length of Unit: 9 weeks

LEARNING TARGETS

NJ Student Learning Standards:

- 8.1.12.AP.1** Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.7** Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
- 8.1.12.AP.8** Evaluate and refine computational artifacts to make them more usable and accessible.
- 8.1.12.AP.9** Collaboratively document and present design decisions in the development of complex programs.
- 8.1.12.CS.4** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

Career Readiness, Life Literacies, and Key Skills:

- 9.3.IT-PRG.4** Demonstrates the effective use of software development tools to develop software applications.
- 9.3.IT-PRG.5** Apply an appropriate software development process to design a software application.
- 9.3.IT-PRG.6** Program a computer application using the appropriate programming language.
- 9.4.12.CI.1** Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
- 9.4.12.CT.1** Identify problem-solving strategies used in the development of an innovative product or practice.

Interdisciplinary Connections and Standards:

ELA

- RI.CI.11–12.2** Determine two or more central ideas of an informational text and analyze how they are developed and refined over the course of a text, including how they interact and build on one another to provide a complex account or analysis; provide an objective summary of the text.
- RI.MF.11–12.6** Synthesize complex information across multiple sources and formats to develop ideas, resolve conflicting information, or develop an interpretation that goes beyond explicit text information (e.g., express a personal point of view, new interpretation of the concept).

SL.II.11–12.2 Integrate multiple sources of information presented in diverse formats and media (e.g., visually, quantitatively, orally) in order to make informed decisions and solve problems, evaluating the credibility and accuracy of each source and noting any discrepancies among the data.

Mathematics

A.SSE.A.1 Interpret expressions that represent a quantity in terms of its context. a. Interpret parts of an expression, such as terms, factors, and coefficients. b. Interpret complicated expressions by viewing one or more of their parts as a single entity.

A.SSE.A.2 Use the structure of an expression to identify ways to rewrite it.

A.SSE.B.3 Choose and produce an equivalent form of an expression to reveal and explain properties of the quantity represented by the expression.

a. Factor a quadratic expression to reveal the zeros of the function it defines.

b. Complete the square in a quadratic expression to reveal the max or min value of the function it defines.

c. Use the properties of exponents to transform expressions for exponential functions.

Unit Understandings:

Students will understand that...

- JavaScript is a real-world programming language that is still used today in many websites and software programs.
- Variables are used to store information inside our programs and help us solve complicated tasks that would otherwise be very difficult without their use.
- User input allows for interaction between the program and the person running the computer program.
- Arithmetic expressions are an integral part of computer programming and its functionality.
- Graphics can be created and modified in JavaScript in a variety of ways.

Unit Essential Questions:

- What is similar in how programs are created in JavaScript versus Karel, and what is distinctly different?
- How do variables change our approach to solving programming tasks?
- How does user input affect our ability to create complex, engaging programs?
- What are graphics in JavaScript, and how do we manipulate their various features?

Knowledge and Skills:

Students will know...

- How to print text on a screen.
- How to prompt a user to enter data (text data, numerical data, etc.).
- How to create a variable, assign it a value, modify its value, and use it within a program.
- What the mathematical operator modulus does, and why it is used in programming.
- How to create and manipulate graphics in JavaScript.
- The fundamental data types in JavaScript (e.g., numbers, strings, booleans).
- How to declare and initialize variables using appropriate keywords (var, let, const).
- The differences between global and local scope for variables.
- The various arithmetic operators (addition, subtraction, multiplication, division) and their order of operations.
- The concept of concatenation is used when combining strings and variables for output.
- The coordinate system used in JavaScript graphics (e.g., origin at top-left).
- Common graphic primitives (e.g., circles, rectangles, lines, text) and their properties (e.g., x, y, radius, width, height, color).
- The process of debugging JavaScript code using console logs and browser developer tools.
- The importance of comments for explaining complex or non-obvious parts of JavaScript code.

Students will be able to...

- Print basic text on a screen to a user running a program.
- Create and use variables to store information in a program.
- Create programs that allow input from the user.
- Use arithmetic expressions to help solve programming tasks.
- Create graphics on a screen in JavaScript.
- Translate programming concepts learned with Karel (functions, loops, conditionals) into JavaScript syntax.
- Define and utilize variables to store data, such as user input, calculations, and graphic properties.
- Design interactive programs that respond to user input (e.g., simple calculators, personalized greetings, games).
- Construct and manipulate various graphic shapes on a canvas, including setting their position, size, and color.
- Combine text output, user input, and graphic elements to create visually engaging and functional programs.
- Perform mathematical calculations using arithmetic operators to solve problems and position graphics accurately.
- Debug JavaScript programs using console output and systematically troubleshoot common errors.
- Collaborate effectively with peers on JavaScript coding projects, providing and incorporating feedback.
- Document their JavaScript code clearly to explain functionality and design choices to others.
- Evaluate and refine their JavaScript programs to improve functionality, usability, and visual appeal.

EVIDENCE OF LEARNING

Assessment:

What evidence will be collected and deemed acceptable to show that students truly “understand”?

- End of Unit Common Assessment - See folder for assessment links.
 - Students will take a test for the unit in which they will have to identify terminology, as well as the key concepts of the unit. Students will also be given a program to solve using programming and explain their thought process for the solution code.
- CodeHS.com lesson exercises
- Class participation
- Class discussion

Learning Activities:

What differentiated learning experiences and instruction will enable all students to achieve the desired results?

- Group programming projects
- Teacher demonstrations over the Screen Share software
- Reinforcement worksheets and extra practice
- Paired-programming challenges
- Individualized student-to-teacher code demonstrations

RESOURCES

Teacher Resources:

- Demonstrations of worked-out solutions from CodeHS.com
- Teacher designed worksheets
- CodeHS.com lesson exercises and videos

Equipment Needed:

- Chromebooks
- Access to high-speed internet
- CodeHS.com login

UNIT 3 OVERVIEW

Content Area: Computer Science

Unit Title: JavaScript Control Structures

Target Course/Grade Level: Exploring Computer Science/Grades 9-12

Unit Summary: In this unit, students will learn how to create the various control structures in JavaScript that they first learned in Karel. With the introduction to variables from Unit 2, there are now several new concepts that students will need to master, including Booleans, Logical Operators, and Comparison Operators. These new topics and concepts play an integral role in the functionality of many of the control structures students have used in Karel (and continue to use it in JavaScript). Students are also introduced to the random function, which allows a user to obtain random numbers, colors, or Booleans from the computer for use in their programs.

Approximate Length of Unit: 9 weeks

LEARNING TARGETS

NJ Student Learning Standards:

- 8.1.12.AP.1** Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.3** Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
- 8.1.12.AP.7** Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
- 8.1.12.AP.8** Evaluate and refine computational artifacts to make them more usable and accessible.
- 8.1.12.AP.9** Collaboratively document and present design decisions in the development of complex programs.
- 8.2.12.ED.1** Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
- 8.2.12.ED.4** Design a product or system that addresses a global problem and document decisions made based on research, constraints, trade-offs, and aesthetic and ethical considerations and share this information with an appropriate audience.
- 8.1.12.CS.4** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

Career Readiness, Life Literacies, and Key Skills:

- 9.3.IT-PRG.4** Demonstrates the effective use of software development tools to develop software applications.
- 9.3.IT-PRG.5** Apply an appropriate software development process to design a software application.
- 9.3.IT-PRG.6** Program a computer application using the appropriate programming language.
- 9.4.12.CI.1** Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
- 9.4.12.CT.1** Identify problem-solving strategies used in the development of an innovative product or practice.

Interdisciplinary Connections and Standards:

ELA

RI.CI.11–12.2 Determine two or more central ideas of an informational text and analyze how they are developed and refined over the course of a text, including how they interact and build on one another to provide a complex account or analysis; provide an objective summary of the text.

RI.MF.11–12.6 Synthesize complex information across multiple sources and formats to develop ideas, resolve conflicting information, or develop an interpretation that goes beyond explicit text information (e.g., express a personal point of view, new interpretation of the concept).

SL.II.11–12.2 Integrate multiple sources of information presented in diverse formats and media (e.g., visually, quantitatively, orally) in order to make informed decisions and solve problems, evaluating the credibility and accuracy of each source and noting any discrepancies among the data.

Mathematics

A.SSE.A.1 Interpret expressions that represent a quantity in terms of its context. a. Interpret parts of an expression, such as terms, factors, and coefficients. b. Interpret complicated expressions by viewing one or more of their parts as a single entity.

A.SSE.A.2 Use the structure of an expression to identify ways to rewrite it.

A.SSE.B.3 Choose and produce an equivalent form of an expression to reveal and explain properties of the quantity represented by the expression.

a. Factor a quadratic expression to reveal the zeros of the function it defines.

b. Complete the square in a quadratic expression to reveal the max or min value of the function it defines.

c. Use the properties of exponents to transform expressions for exponential functions.

Unit Understandings:

Students will understand that...

- Booleans are a primitive form of data that can be read directly by a computer.
- Comparison and Logical operators are an integral part of the control structures of programming.
- Nested control structures add an even greater level of complexity to programming.

Unit Essential Questions:

- What is a Boolean, and how is it used in JavaScript?
- How do Comparison and Logical Operators make the various control structures work?
- How do you decide when and how to nest control structures when programming?

Knowledge and Skills:

Students will know...

- Booleans can take on either a true or false value.
- Logical operators are used to combine Boolean expressions and are used in if/if-else statements and while loops.
- Comparison operators are used for mathematical expressions and in loop structures.
- Control structures help solve large and complex problems in programming.
- The specific JavaScript syntax for if, if-else, for, and while statements, including the use of parentheses and curly braces.
- The difference between definite and indefinite loops and when to use for vs. while.
- The truth tables for logical operators (&& (AND), || (OR), ! (NOT)) and how they evaluate compound conditions.
- The various comparison operators (==, ===, !=, !==, >, <, >=, <=) and their role in evaluating conditions.
- The difference between equality (==) and strict equality (===) in JavaScript and when to use each.

- How the Math.random() function works and how to use it to generate random numbers within a specific range.
- How to generate random integers, floating-point numbers, or select random elements (like colors) using Math.random() in combination with other arithmetic.
- The concept of "flow of control" in a program and how control structures alter this flow.
- Effective debugging strategies are essential when dealing with complex conditional logic and loops.

Students will be able to...

- Create and use Boolean variables and expressions.
- Determine whether a combination of Boolean expressions is true or false based on the truth tables for the Logical Operators AND or OR.
- Use Comparison Operators to solve programming tasks.
- Create and use If/If-Else Statements in JavaScript.
- Create and use For Loops in JavaScript.
- Use the Random function to solve programming tasks.
- Create and use While Loops in JavaScript.
- Construct if, if-else, and nested if-else statements to execute different code blocks based on specified conditions.
- Write for loops to repeat a block of code a predetermined number of times, managing iteration variables effectively.
- Implement while loops to repeat code as long as a certain condition remains true, ensuring proper termination conditions.
- Combine Boolean expressions using logical operators to create sophisticated conditions for control structures.
- Apply comparison operators within if statements and loop conditions to compare values and make decisions.
- Design and implement programs that involve random elements, such as games, simulations, or dynamic graphics.
- Select the most appropriate control structure (if, if-else, for loop, while loop) for a given programming problem.
- Trace the execution of JavaScript code involving nested control structures and complex Boolean logic to predict outcomes.
- Systematically debug programs containing errors in conditional logic, loop termination, or random number generation.
- Collaborate and document their design decisions for programs that leverage advanced control structures, considering readability and maintainability.

EVIDENCE OF LEARNING

Assessment:

What evidence will be collected and deemed acceptable to show that students truly "understand"?

- End of Unit Common Assessment - See folder for assessment links.
 - Students will take a test for the unit in which they will have to identify terminology, as well as the key concepts of the unit. Students will also be given a program to solve using programming and explain their thought process for the solution code.
- CodeHS.com lesson exercises

- Class participation
- Class discussion

Learning Activities:

What differentiated learning experiences and instruction will enable all students to achieve the desired results?

- Group programming projects
- Teacher demonstrations over the Screen Share software
- Reinforcement worksheets and extra practice
- Paired-programming challenges
- Individualized student-to-teacher code demonstrations

<i>RESOURCES</i>

Teacher Resources:

- Demonstrations of worked-out solutions from CodeHS.com
- Teacher designed worksheets
- CodeHS.com lesson exercises and videos

Equipment Needed:

- Chromebooks
- Access to high-speed internet
- CodeHS.com login

UNIT 4 OVERVIEW

Content Area: Computer Science

Unit Title: Functions and Parameters

Target Course/Grade Level: Exploring Computer Science/Grades 9-12

Unit Summary: In this unit, students will learn how to create and use functions in JavaScript. Students learned what a function was and how to both define and call them with Karel. Functions in JavaScript are similar, but with one key difference: the inclusion of parameters. Students will learn what parameters are and how to send them along while calling a function. Also, students will learn how functions with return values work and why return values are preferred in certain situations.

Approximate Length of Unit: 8 weeks

LEARNING TARGETS

NJ Student Learning Standards:

- 8.1.12.AP.1** Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.2** Create generalized computational solutions using collections instead of repeatedly using simple variables.
- 8.1.12.AP.3** Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
- 8.1.12.AP.5** Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 8.1.12.AP.6** Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 8.1.12.AP.7** Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
- 8.1.12.AP.8** Evaluate and refine computational artifacts to make them more usable and accessible.
- 8.1.12.AP.9** Collaboratively document and present design decisions in the development of complex programs.
- 8.2.12.ED.1** Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
- 8.2.12.ED.4** Design a product or system that addresses a global problem and document decisions made based on research, constraints, trade-offs, and aesthetic and ethical considerations and share this information with an appropriate audience.
- 8.1.12.CS.4** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- 8.2.12.NT.2** Redesign an existing product to improve form or function.

Career Readiness, Life Literacies, and Key Skills:

- 9.3.IT-PRG.4** Demonstrates the effective use of software development tools to develop software applications.

- 9.3.IT-PRG.5** Apply an appropriate software development process to design a software application.
- 9.3.IT-PRG.6** Program a computer application using the appropriate programming language.
- 9.4.12.CI.1** Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
- 9.4.12.CT.1** Identify problem-solving strategies used in the development of an innovative product or practice.

Interdisciplinary Connections and Standards:

ELA

- RI.CI.11–12.2** Determine two or more central ideas of an informational text and analyze how they are developed and refined over the course of a text, including how they interact and build on one another to provide a complex account or analysis; provide an objective summary of the text.
- RI.MF.11–12.6** Synthesize complex information across multiple sources and formats to develop ideas, resolve conflicting information, or develop an interpretation that goes beyond explicit text information (e.g., express a personal point of view, new interpretation of the concept).
- SL.II.11–12.2** Integrate multiple sources of information presented in diverse formats and media (e.g., visually, quantitatively, orally) in order to make informed decisions and solve problems, evaluating the credibility and accuracy of each source and noting any discrepancies among the data.

Mathematics

- A.SSE.A.1** Interpret expressions that represent a quantity in terms of its context. a. Interpret parts of an expression, such as terms, factors, and coefficients. b. Interpret complicated expressions by viewing one or more of their parts as a single entity.
- A.SSE.A.2** Use the structure of an expression to identify ways to rewrite it.
- A.SSE.B.3** Choose and produce an equivalent form of an expression to reveal and explain properties of the quantity represented by the expression.
- Factor a quadratic expression to reveal the zeros of the function it defines.
 - Complete the square in a quadratic expression to reveal the max or min value of the function it defines.
 - Use the properties of exponents to transform expressions for exponential functions.

Unit Understandings:

Students will understand that...

- Functions are essential in higher-level programming, both in Karel and in JavaScript (and beyond).
- Parameters are values sent to a function when called to allow the function to perform some action or computation based on the value of those parameters.
- Functions with return values are useful in certain scenarios and are sometimes preferred to functions without return values.

Unit Essential Questions:

- Why do we use functions in JavaScript?
- What is the advantage of creating a function in computer programming?
- When would we use a function with a return value over a function without a return value, and vice versa?

Knowledge and Skills:

Students will know...

- How to define and call a function in JavaScript.
- What a parameter is.
- What a return value is, and how it is used.
- Why functions are essential to programming.
- The syntax for defining functions with and without parameters in JavaScript.
- The syntax for calling functions and passing arguments (values for parameters).

- The concept of scope as it applies to variables declared inside functions (local variables) versus outside functions (global variables).
- How parameters act as placeholders for values that are passed into a function.
- The difference between a parameter (in the function definition) and an argument (in the function call).
- The syntax for using the return keyword to send a value back from a function.
- When a function's purpose necessitates a return value (e.g., calculation, validation) versus when it performs an action (e.g., drawing graphics, printing).
- The benefits of using functions for code organization, reusability, and reducing redundancy (DRY principle - Don't Repeat Yourself).
- How functions promote problem decomposition, breaking complex problems into smaller, more manageable sub-problems.
- The concept of function signature (function name and its parameters).

Students will be able to...

- Define and call functions in JavaScript.
- Create programs that use functions, both with and without return values.
- Define JavaScript functions that accept one or more parameters to make them more versatile.
- Call functions and correctly pass the appropriate number and type of arguments.
- Write functions that return values (e.g., calculations, boolean checks) to be used in other parts of the program.
- Integrate functions with control structures (conditionals, loops) to create modular and dynamic programs.
- Decompose larger programming tasks into a set of well-defined, reusable functions.
- Explain the flow of execution when functions are called, including how arguments are passed and return values are handled.
- Choose between using a global variable and passing a parameter based on the desired scope and reusability of data.
- Refactor existing code to incorporate functions, improving its readability, maintainability, and efficiency.
- Collaborate effectively on programming projects, designing and implementing functions collectively.
- Debug programs that involve function calls, parameter passing, and return values, understanding common errors like undefined variables or incorrect argument counts.
- Document their functions with comments explaining their purpose, parameters, and return values to enhance code readability for others.

EVIDENCE OF LEARNING

Assessment:

What evidence will be collected and deemed acceptable to show that students truly “understand”?

- End of Unit Common Assessment - See folder for assessment links.
 - Students will take a test for the unit in which they will have to identify terminology, as well as the key concepts of the unit. Students will also be given a program to solve using programming and explain their thought process for the solution code.
- CodeHS.com lesson exercises
- Class participation

- Class discussion

Learning Activities:

What differentiated learning experiences and instruction will enable all students to achieve the desired results?

- Group programming projects
- Teacher demonstrations over the Screen Share software
- Reinforcement worksheets and extra practice
- Paired-programming challenges
- Individualized student-to-teacher code demonstrations

RESOURCES

Teacher Resources:

- Demonstrations of worked-out solutions from CodeHS.com
- Teacher designed worksheets
- CodeHS.com lesson exercises and videos

Equipment Needed:

- Chromebooks
- Access to high-speed internet
- CodeHS.com login

UNIT 5 OVERVIEW

Content Area: Computer Science

Unit Title: Animation and Games

Target Course/Grade Level: Exploring Computer Science/Grades 9-12

Unit Summary: In this unit, students will learn basic animation functions and create simple yet satisfying computer games from scratch. Students will learn how to use the timer function, mouse events, and keyboard events to create interactive computer games in JavaScript. The unit culminates in a Final Project to create the classic game Breakout. Students will work together in groups of 2-3 to build each part of the Breakout game, and come together at the end to put each piece together in one full program.

Approximate Length of Unit: 6 weeks

LEARNING TARGETS

NJ Student Learning Standards:

- 8.1.12.AP.1** Design algorithms to solve computational problems using a combination of original and existing algorithms.
- 8.1.12.AP.2** Create generalized computational solutions using collections instead of repeatedly using simple variables.
- 8.1.12.AP.3** Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
- 8.1.12.AP.4** Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
- 8.1.12.AP.5** Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 8.1.12.AP.6** Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 8.1.12.AP.7** Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
- 8.1.12.AP.8** Evaluate and refine computational artifacts to make them more usable and accessible.
- 8.1.12.AP.9** Collaboratively document and present design decisions in the development of complex programs.
- 8.2.12.ED.1** Use research to design and create a product or system that addresses a problem and make modifications based on input from potential consumers.
- 8.2.12.ED.4** Design a product or system that addresses a global problem and document decisions made based on research, constraints, trade-offs, and aesthetic and ethical considerations and share this information with an appropriate audience.
- 8.1.12.CS.4** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- 8.2.12.NT.2** Redesign an existing product to improve form or function.

Career Readiness, Life Literacies, and Key Skills:

- 9.3.IT-PRG.4** Demonstrates the effective use of software development tools to develop software applications.

- 9.3.IT-PRG.5** Apply an appropriate software development process to design a software application.
- 9.3.IT-PRG.6** Program a computer application using the appropriate programming language.
- 9.4.12.CI.1** Demonstrate the ability to reflect, analyze, and use creative skills and ideas.
- 9.4.12.CT.1** Identify problem-solving strategies used in the development of an innovative product or practice.

Interdisciplinary Connections and Standards:

ELA

- RI.CI.11–12.2** Determine two or more central ideas of an informational text and analyze how they are developed and refined over the course of a text, including how they interact and build on one another to provide a complex account or analysis; provide an objective summary of the text.
- RI.MF.11–12.6** Synthesize complex information across multiple sources and formats to develop ideas, resolve conflicting information, or develop an interpretation that goes beyond explicit text information (e.g., express a personal point of view, new interpretation of the concept).
- SL.II.11–12.2** Integrate multiple sources of information presented in diverse formats and media (e.g., visually, quantitatively, orally) in order to make informed decisions and solve problems, evaluating the credibility and accuracy of each source and noting any discrepancies among the data.

Mathematics

- A.SSE.A.1** Interpret expressions that represent a quantity in terms of its context. a. Interpret parts of an expression, such as terms, factors, and coefficients. b. Interpret complicated expressions by viewing one or more of their parts as a single entity.
- A.SSE.A.2** Use the structure of an expression to identify ways to rewrite it.
- A.SSE.B.3** Choose and produce an equivalent form of an expression to reveal and explain properties of the quantity represented by the expression.
- Factor a quadratic expression to reveal the zeros of the function it defines.
 - Complete the square in a quadratic expression to reveal the max or min value of the function it defines.
 - Use the properties of exponents to transform expressions for exponential functions.

Unit Understandings:

Students will understand that...

- JavaScript can be used to create animations and computer games.
- Computer games are highly complex computer programs that involve high-level functions and problem-solving.
- Creating computer games is challenging and requires adaptive and novel approaches in order to make the game work correctly and run smoothly.

Unit Essential Questions:

- What is the goal of creating a computer game?
- How does one approach making a game mechanic work, and work well?
- How are tasks to be divided when creating a large, complex program such as a computer game?
- What is troubleshooting, debugging, and pseudocode, and is it used/important?
- How do people develop, test, and debug programs?

Knowledge and Skills:

Students will know...

- How to program using Events (e.g. clicking a mouse, hitting “enter” on the keyboard).
- How to use timers and randomizers to make certain features of a game work.
- What debugging is, and how to go about debugging.
- What pseudocode is, and how it is used.
- Computer games are complex, multi-faceted programs that involve unique approaches to problems.

- Project programming is highly involved and requires many different skill sets.
- The concept of an event-driven program and how it differs from sequential program execution.
- How event listeners (e.g., onmousedown, onkeydown) are used to detect user interactions.
- The relationship between frames per second (FPS) and smooth animation.
- The concept of game loops and their role in continuously updating game state and rendering graphics.
- Basic collision detection principles for determining when game objects interact.
- The importance of state variables to track game progress (e.g., score, lives, game over).
- Common game development patterns such as updating object positions, drawing objects, and handling input.
- Strategies for breaking down a large project (like Breakout) into smaller, manageable components for group work.
- The importance of clear communication and version control in collaborative programming projects.
- The iterative nature of game development, involving continuous testing, debugging, and refinement.
- The role of pseudocode and flowcharts in planning complex game logic.

Students will be able to...

- Use timers to create animations.
- Use mouse events to allow user interactions in a program/game.
- Use keyboard events to allow user interactions in a program/game.
- Create the game Breakout by working in a group and dividing up the various aspects of the game, where each student works on a part of the game, and then combines all of the work into one final program.
- Implement event handlers for mouse clicks, mouse movements, and keyboard presses to control game elements.
- Utilize setInterval (or equivalent animation functions) to create continuous movement and animations for game objects.
- Design and implement collision detection logic between various game elements (e.g., ball and paddle, ball and bricks, ball and walls).
- Manage game state using variables to track score, lives, levels, and win/loss conditions.
- Develop functions and modular code to represent different game components (e.g., creating bricks, moving the ball, handling paddle input).
- Collaborate effectively in a group setting to divide tasks, integrate code, and troubleshoot problems in a shared project.
- Design, test, and refine game mechanics iteratively based on user feedback and playtesting.
- Use debugging techniques specific to animation and interactive programs, such as logging variable values or observing object behavior.
- Create clear documentation for their game's code, explaining how different components work and how they integrate.
- Present their game and explain their design choices, problem-solving strategies, and the collaborative process.
- Analyze and adapt existing algorithms to fit specific game mechanics (e.g., bouncing logic).

EVIDENCE OF LEARNING

Assessment:

What evidence will be collected and deemed acceptable to show that students truly “understand”?

- End of Unit Common Assessment - See folder for assessment links.
 - Students will take a test for the unit in which they will have to identify terminology, as well as the key concepts of the unit. Students will also be given a program to solve using programming and explain their thought process for the solution code.
- CodeHS.com lesson exercises
- Class participation
- Class discussion

Learning Activities:

What differentiated learning experiences and instruction will enable all students to achieve the desired results?

- Group programming projects
- Teacher demonstrations over the Screen Share software
- Reinforcement worksheets and extra practice
- Paired-programming challenges
- Individualized student-to-teacher code demonstrations

RESOURCES

Teacher Resources:

- Demonstrations of worked-out solutions from CodeHS.com
- Teacher designed worksheets
- CodeHS.com lesson exercises and videos

Equipment Needed:

- Chromebooks
- Access to high-speed internet
- CodeHS.com login