Wall Township Public Schools	Board Approval Date:

Course Title: INTRODUCTION TO COMPUTER SCIENCE IN JAVASCRIPT & VIDEO GAME DESIGN

Course Description: This course fulfills the graduation requirement for career education/practical arts. This introduction to computer science and video game design curriculum teaches the foundations of computer science and basic programming, with an emphasis on helping students develop the logical thinking and problem-solving skills required to code video games. The Intro to Computer Science and Video Game Design course is designed for complete beginners with no previous background in computer science, but does provide opportunities to learn advanced topics. The course is highly visual, dynamic, and interactive, making it engaging for new coders. Topics covered include JavaScript basics, control structures, graphics, animations, and more.

New Jersey Student Learning Standards
• 2020 New Jersey Student Learning Standards – Computer Science and Design Thinking Introduction
• 2020 New Jersey Student Learning Standards – Career Readiness, Life Literacies, and Key Skills

Course Sequence & Pacing	
Unit Title	**Estimated Number of Instructional Periods**
Unit 1: Intro to Programming with Karel	18
Unit 2: JavaScript Basics	4
Unit 3: JavaScript Graphics	8
Unit 4: JavaScript Control Structures	15
Unit 5: JavaScript Functions	12
Unit 6: Animations	28
Unit 7: Data Structures	22
Unit 8: Creating JavaScript Games	28

Unit 1

Desired Results
Unit 1 Title: Intro to Programming with Karel
Unit Summary:

In this unit, students learn foundational programming concepts using Karel the Dog, a virtual robot in a grid world. They write JavaScript programs to command Karel to perform tasks, developing skills in commands, method creation, control flow, loops, and conditionals. Students practice breaking down complex problems into smaller parts with top-down design, learn how to document code with comments, and apply preconditions and postconditions to their programs. Through a series of programming exercises and challenges, students gain hands-on experience that builds problem-solving and logical thinking skills fundamental to programming.

Learning Standards

NJSLS:

- 8.1.12.CS.4: Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- 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.
- 8.1.12.AP.7: Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
- 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.NT.1: Explain how different groups can contribute to the overall design of a product.
- 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.
- 9.4.12.CT.2: Explain the potential benefits of collaborating to enhance critical thinking and problem solving.
- 9.4.12.TL.1: Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task.
- 9.4.12.TL.3: Analyze the effectiveness of the process and quality of collaborative environments.

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 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.
- 9.4.12.CT.2: Explain the potential benefits of collaborating to enhance critical thinking and problem solving.
- 9.4.12.TL.1: Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task.
- 9.4.12.TL.3: Analyze the effectiveness of the process and quality of collaborative environments.

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop a model based on evidence to illustrate the relationships between systems or between components of a system.
- Use a model to predict the relationships between systems or between components of a system.
- Plan and conduct an investigation individually and collaboratively to produce data to serve as the basis for evidence, and in the design: decide on types, how much, and accuracy of data needed to produce reliable measurements and consider limitations on the precision of the data (e.g., number of trials, cost, risk, time), and refine the design accordingly.
- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Apply scientific ideas to solve a design problem, taking into account possible unanticipated effects.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.

- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Apply scientific reasoning to link evidence to the claims to assess the extent to which the reasoning and data support the explanation or conclusion.
- Construct an explanation based on valid and reliable evidence obtained from a variety of sources (including students' own investigations, models, theories, simulations, peer review) and the assumption that theories and laws that describe the natural world operate today as they did in the past and will continue to do so in the future.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-3 Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:	**Unit Enduring Understandings:**
• How do we translate real-world problems into step-by-step instructions for a computer? • What strategies can we use to break down complex programming tasks?	• Programming allows us to give precise instructions to a computer to solve problems. • Complex problems can be solved more effectively by breaking them down into smaller, manageable subproblems.

• How do loops and conditionals control the flow of a program? • Why is it important to write clear comments and define preconditions/postconditions in code? • How do methods help in organizing and simplifying code?	• Control flow structures such as loops and conditionals enable programs to make decisions and repeat actions. • Writing clear, commented code improves program readability and maintainability. • Preconditions and postconditions help define expectations and verify program correctness. • Using methods (defining and calling) supports code reuse and organization.
Vocabulary: Commands, Method, Control Flow, Loop, Conditional, Comment, Precondition, Postcondition, Top Down Design, Program Entry Point	
Students will be able to: • Identify and brainstorm ways computers have impacted their own lives, as well as society, economy, and culture • Develop their own learning objectives for the course by reflecting on why they are taking this course • Recognize, explain, and use the commands that Karel can be given. These commands are: move(), putBall(), takeBall() and turnLeft() • Recognize, explain, and use the commands that Karel can be given • Debug and rework their code as well as someone else's code • Define and call functions • Understand what functions are, how they are used and how using them improves programs • Design and implement their own functions to solve problems • Explain the importance of writing readable code • Analyze and compare the readability of different programs • Use the start function to make their programs more readable • Break a large problem down into smaller, simpler problems • Write functions that solve the simpler problems, and use them as building blocks to solve the larger problem	

- Compare programs and identify good vs. poor decomposition
- Explain the preconditions and postconditions of a function
- Create clear and readable comments in their code that help the reader understand the code
- Explain the purpose of comments
- Understand abstraction as the different levels of detail and complexity
- Understand the importance of abstracting away complexity to solve problems more efficiently
- Write programs that use SuperKarel instead of Karel
- Utilize the new toolbox of commands that SuperKarel provides over Karel
- Read the documentation to understand how to use an API (SuperKarel is an example of this)
- Create for loops to repeat code a fixed number of times
- Explain when a for loop would be a useful tool
- Utilize for loops to write programs that would be difficult/impossible without loops
- Use conditions to gather information about Karel's world (is the front clear, is Karel facing north, etc)
- Create if statements to only execute code if a certain condition is true
- Explain the purpose of an If/Else statement
- Create If/Else statements to solve new types of problems
- Identify when an If/Else statement is appropriate to be used
- Explain the purpose of a while loop
- Create while loops to repeat code while a condition is true
- Utilize while loops to solve new types of problems
- Test their solutions on different Karel worlds
- Identify the different control structures that can be used to modify the flow of control through a program
- Combine control structures to solve complicated problems
- Choose the proper control structure for a given problem
- Students will be able to use debugging strategies to find and fix errors in their code.
- Analyze an algorithm and explain why it works
- Use control structures to create general algorithms that work on all Karel worlds

- Explain why it is important to indent code
- Identify proper indentation
- Modify a program to have proper indentation
- Write programs with proper indentation

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive Karel programming exercises progressing from basic commands to complex problem-solving
- Pair programming sessions where students collaborate on Karel challenges
- Code debugging workshops using peer review and systematic debugging strategies
- Function design workshops with real-world problem decomposition exercises
- Algorithm analysis activities comparing different solutions to the same Karel problem
- Creative expression projects where students design their own Karel worlds and solutions

Differentiation Strategies:

- For Advanced Learners: Provide extension challenges with multiple solution paths, mentor struggling peers, create original Karel worlds with custom challenges
- For Struggling Learners: Offer step-by-step guided practice, visual flowcharts for algorithm planning, one-on-one coding support sessions
- For English Language Learners: Use visual programming aids, provide glossary of programming terms, allow code comments in native language initially
- Universal Design for Learning: Multiple representation formats (visual, kinesthetic, auditory), choice in final project topics, flexible pacing for concept mastery

Education Connections:

- Highlight diverse programmers and computer scientists in historical contexts

- Discuss how programming accessibility features help people with disabilities
- Explore how computational thinking applies across cultures and communities
- Address equity in technology access and opportunities

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Random Hurdles

      Write a program that has Karel run to the other side of first street, jumping over all of the hurdles. However, the hurdles can be in random locations. The world is fourteen avenues long.
    - Example Exercise: Super Cleanup Karel

      Karel's world is a complete mess. There are tennis balls all over the place, and you need to clean them up. Karel will start in the bottom left corner of the world facing east, and should clean up all of the tennis balls in the world. This program should be general enough to work on any size world with tennis balls in any locations.
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Karel Challenges

Common Benchmark Assessments:

- Karel Challenges

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- CodeHS Karel programming platform and curriculum
- Karel JavaScript library and development environment
- Interactive debugging tools and error analysis resources
- Function documentation and API reference guides

Supplemental Materials:

- Visual flowchart templates for algorithm design
- Peer collaboration rubrics for pair programming
- Real-world problem-solving case studies
- Assessment rubrics for creative projects

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Additional time - Review of directions - Space for movement or breaks - Extra visual and verbal cues and prompts - Follow a routine/schedule - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Visual learning, including graphic organizers - Teacher modeling - Pairing students with beginning English language skills with students who have more advanced English language skills - Think-pair-share - Cooperative learning groups

Students at Risk of Failure

- Alternative Assessments - Learning Contracts - Supplemental Instruction - Think-Pair-Share by Readiness - Differentiated Online Practice - Tiered Level of Supports - Tiered Activities - Small Group Instruction - Work Alone Choice - Group Work Choice

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students

It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

- Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments - Pre-Assess to provide Differentiated Instruction to Accelerate Learning

Unit 2

Desired Results

Unit 2 Title: JavaScript Basics

Unit Summary:

In this unit, students learn the basics of JavaScript, including variables, user input, arithmetic expressions, constants, random numbers, and functions. Through interactive exercises, they write simple programs that collect input, perform calculations, and respond with personalized output. Students also practice collaborative programming and begin developing clear, functional code that demonstrates core problem-solving and logical thinking skills.

Learning Standards

NJSLS:

- 8.1.12.CS.1 - Describe ways in which integrated systems hide underlying implementation details to simplify user experiences
- 8.1.12.CS.4 - Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors
- 8.1.12.DA.3 - Translate between decimal numbers and binary numbers
- 8.1.12.DA.4 - Explain the relationship between binary numbers and the storage and use of data in a computing device
- 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
- 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.IC.1 - Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices
- 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

Interdisciplinary Connections
Career Readiness, Life Literacies, & Key Skills *(CLKS)*: ● 9.4.12.CT.1 - Identify problem-solving strategies used in the development of an innovative product or practice ● 9.4.12.CT.2 - Explain the potential benefits of collaborating to enhance critical thinking and problem solving ● 9.4.12.DC.6 - Select information to post online that positively impacts personal image and future college and career opportunities ● 9.4.12.DC.7 - Evaluate the influence of digital communities on the nature, content and responsibilities of careers ● 9.4.12.IML.1 - Compare search browsers and recognize features that allow for filtering of information ● 9.4.12.IML.2 - Evaluate digital sources for timeliness, accuracy, perspective, credibility of the source, and relevance of information ● 9.4.12.IML.3 - Analyze data using tools and models to make valid and reliable claims ● 9.4.12.TL.1 - Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task ● 9.4.12.TL.2 - Generate data using formula-based calculations in a spreadsheet and draw conclusions about the data ● 9.4.12.TL.3 - Analyze the effectiveness of the process and quality of collaborative environments ● 9.4.12.CI.1 - Demonstrate the ability to reflect, analyze, and use creative skills and ideas ● 9.4.12.CI.2 - Identify career pathways that highlight personal talents, skills, and abilities
Interdisciplinary Connections: *Science and Engineering Practices:* ● Develop a model based on evidence to illustrate the relationships between systems or between components of a system. ● Use a model to predict the relationships between systems or between components of a system. ● Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution. ● Use mathematical and/or computational representations of phenomena or design solutions to support explanations.

- Apply scientific ideas to solve a design problem, taking into account possible unanticipated effects.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.
- Apply scientific reasoning to link evidence to the claims to assess the extent to which the reasoning and data support the explanation or conclusion.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How can we use variables and user input to make programs interactive and personalized?
- Why are functions and clear code structure important when building and maintaining programs?

Unit Enduring Understandings:

- Programs become more dynamic and useful when they can store data and respond to user input.
- Using functions and clear code structure helps programmers write reusable, organized, and maintainable code.

Vocabulary: Define a Function, Call a Function, Variable, Constant, Argument, Parameter, Return, Return Value

Students will be able to:

- Write a JavaScript program by typing commands with proper syntax in the main function
- Write a program that prints out a message to the user

- Explain what variables are and what they are used for
- Create their own variables
- Print out the values stored in variables
- Create programs that ask the user for input
- Store user input in variables and print it back to the user
- Choose the proper input function to use depending on the type of information needed
- Describe the different mathematical operators we can use in programs
- Create programs that use basic math to compute useful things
- Create programs that take in user input, do simple computations with the input, and produce useful output
- Effectively communicate their ideas to a partner
- Successfully complete a coding exercise using pair programming
- Identify the pros and cons of pair programming
- Explain why random numbers are a useful part of computer programs.
- Create random values in a program.
- Utilize the DOCS for the Randomizer class in order to learn how to generate random values.
- Define JavaScript functions
- Call JavaScript functions within the main function
- Use functions in order to manage the flow of their programs
- Increase the readability and organization of their code using functions

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Hands-on JavaScript exercises focused on variables, user input, math operations, and functions
- Pair programming sessions to encourage collaboration and peer learning
- Guided coding labs that walk students through creating interactive programs like personalized greetings
- Exploration of random number generation through real-world examples (e.g., games or simulations)

- Code review and debugging workshops emphasizing clear communication and logical problem solving
- Creative projects where students design simple programs that reflect their interests or cultural experiences

Differentiation Strategies:

- For Advanced Learners: Offer open-ended challenges involving multiple functions and complex input/output, encourage mentoring peers, and support creation of original interactive programs
- For Struggling Learners: Provide scaffolded tutorials, use visual aids like flowcharts or pseudocode, and offer small-group or one-on-one coding support
- For English Language Learners: Incorporate bilingual glossaries of programming terms, allow use of native language in code comments, and use visual programming examples to reinforce concepts
- Universal Design for Learning (UDL): Present concepts through multiple modalities (video, text, hands-on coding), allow choice in program topics, and offer flexible pacing and checkpoints

Education Connections:

- Highlight contributions of diverse programmers and technologists, including Asian American & Pacific Islander pioneers
- Discuss how accessible programming tools support people with disabilities
- Explore how coding skills empower underrepresented communities and can address climate change and social justice issues
- Promote equity by acknowledging barriers in technology access and encouraging inclusive participation

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Dinner Plans

      Prompt the user for their name, then ask them what time you should meet for dinner. Greet them by name and tell them you will meet them at the time they specified!

- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Digital Business Card

Common Benchmark Assessments:

- Digital Business Card

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment
- Interactive JavaScript exercises focused on variables, user input, math operations, and functions
- Debugging tools and error analysis resources within the coding platform
- JavaScript function documentation and API reference guides (including Randomizer class)

Supplemental Materials:

- Visual flowchart and pseudocode templates to plan program logic
- Peer collaboration rubrics and guidelines for effective pair programming
- Real-world problem scenarios to apply JavaScript basics
- Assessment rubrics for coding exercises and creative programming projects

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Review of directions - Student restates information - Space for movement or breaks - Follow a routine/schedule - Rest breaks - Verbal and visual cues regarding directions and staying on task

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Simplified/Verbal Instructions - Frequent Breaks - Teacher modeling

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Learning Contracts - Homogenous Grouping - Supplemental Instruction - Think-Pair-Share by Readiness - Leveled Rubrics

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students

It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

-Homogenous Grouping - Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments

Unit 3

Desired Results

Unit 3 Title: JavaScript Graphics

Unit Summary:

In this unit, students build on their foundational JavaScript skills by exploring the Canvas and Graphics library. They learn how to add and position graphical objects, such as shapes, images, and text, using code. Through creative exercises like designing a custom meme, students practice manipulating visual elements and combining them with code-based logic. In the following unit, students apply their knowledge in more complex, open-ended graphics challenges, such as creating a scalable ghost using constants. These activities emphasize design, precision, and problem-solving, reinforcing both visual creativity and programming structure.

Learning Standards

NJSLS:

- 8.1.12.CS.1 - Describe ways in which integrated systems hide underlying implementation details to simplify user experiences
- 8.1.12.CS.4 - Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors
- 8.1.12.DA.3 - Translate between decimal numbers and binary numbers
- 8.1.12.DA.4 - Explain the relationship between binary numbers and the storage and use of data in a computing device
- 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
- 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.IC.1 - Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices
- 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

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CT.1 - Identify problem-solving strategies used in the development of an innovative product or practice
- 9.4.12.CT.2 - Explain the potential benefits of collaborating to enhance critical thinking and problem solving
- 9.4.12.DC.6 - Select information to post online that positively impacts personal image and future college and career opportunities
- 9.4.12.DC.7 - Evaluate the influence of digital communities on the nature, content and responsibilities of careers
- 9.4.12.IML.1 - Compare search browsers and recognize features that allow for filtering of information
- 9.4.12.IML.2 - Evaluate digital sources for timeliness, accuracy, perspective, credibility of the source, and relevance of information
- 9.4.12.IML.3 - Analyze data using tools and models to make valid and reliable claims
- 9.4.12.TL.1 - Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task
- 9.4.12.TL.2 - Generate data using formula-based calculations in a spreadsheet and draw conclusions about the data
- 9.4.12.TL.3 - Analyze the effectiveness of the process and quality of collaborative environments
- 9.4.12.CI.1 - Demonstrate the ability to reflect, analyze, and use creative skills and ideas
- 9.4.12.CI.2 - Identify career pathways that highlight personal talents, skills, and abilities

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop a model based on evidence to illustrate the relationships between systems or between components of a system.
- Use a model to predict the relationships between systems or between components of a system.
- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.
- Apply scientific ideas to solve a design problem, taking into account possible unanticipated effects.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:	**Unit Enduring Understandings:**
• How can code be used to create and manipulate visual elements on a canvas?	

• In what ways does using constants and positioning enhance the flexibility and scalability of graphic-based programs? • How can solving visual programming challenges help deepen understanding of computational thinking and problem-solving?	• Graphics programming allows developers to create visually engaging programs by combining logic, positioning, and creativity. • Using constants and structured code helps make programs more adaptable and easier to maintain or modify. • Applying programming concepts to visual challenges reinforces critical thinking, debugging skills, and a deeper understanding of code behavior and design.
Vocabulary: Constant, Canvas, Coordinate system, Radius, Variable, Argument, Parameter, Function, Call a Function, Local variable, Global variable, For Loop	
Students will be able to: • Understand the coordinate system of the canvas • Create basic shapes like circles and rectangles • Position shapes in specific locations on the canvas • Learn how to add color to shapes • Understand how the debug mode functions and how to turn it on or off • Add images to their graphics projects using WebImage • Resize image objects using setSize • Display text on the canvas • Break their code into functions based on objects to be rendered • Strategically position shapes anywhere on the canvas • Break their graphics projects into manageable functions • Order their function calls in the main function correctly • Define a problem in their own words and plan out a solution to the problem	

- Break a large problem down into smaller pieces and solve each of the pieces, then use these solutions as building blocks to solve the larger problem
- Write clear and readable graphics programs

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive coding exercises to practice placing and customizing graphic elements using the JavaScript Canvas
- Step-by-step tutorials guiding students in creating personalized projects like memes with images and text
- Creative design challenges that integrate graphics with logic (e.g., drawing characters or symbols with constants)
- Pair programming to support collaboration, problem-solving, and design discussions
- Whole-class debugging sessions where students analyze graphic glitches and propose fixes together
- Reflective prompts encouraging students to use graphic projects to express personal or cultural identity

Differentiation Strategies:

- For Advanced Learners: Introduce complex visual challenges requiring layered graphics, scaling with constants, and interactivity (e.g., animations or mouse events); encourage independent project development
- For Struggling Learners: Provide drag-and-drop visual guides, pre-filled code templates, and focus on step-by-step graphic construction with teacher check-ins
- For English Language Learners: Use labeled visual examples, bilingual vocabulary cards for graphics terminology, and encourage expression through imagery before text
- Universal Design for Learning (UDL): Offer multiple ways to demonstrate understanding (e.g., drawing on paper first, narrating code logic), allow for pacing choices, and support with visuals, videos, and manipulatives

Education Connections:

- Recognize the artistic and technical contributions of Asian American & Pacific Islander developers in digital media
- Explore how accessible graphic tools enable creative expression for people with disabilities

- Encourage students to create visual messages about climate change, equity, or cultural pride through their graphics
- Promote inclusive representation in programming by showcasing diverse characters and themes in student-created images

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Create Your Own Meme

      In this exercise, you are going to create your own meme! The only requirements are that you add at least one image and one text element.
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Graphics Challenges

Common Benchmark Assessments:

- Graphics Challenges

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment with Canvas graphics support
- Interactive exercises focused on drawing, positioning, and styling graphical elements

- Built-in debugging tools and visual feedback for real-time error analysis
- JavaScript graphics documentation and API reference guides (e.g., drawImage, drawText, coordinate system, constants)

Supplemental Materials:

- Visual planning tools such as canvas grid templates, flowcharts, and pseudocode organizers
- Peer collaboration rubrics and pair programming strategies for effective teamwork
- Creative design prompts to apply graphics concepts
- Assessment rubrics for evaluating accuracy, creativity, code structure, and use of constants
- Culturally responsive project examples to inspire inclusive and expressive visuals
- Anchor charts and vocabulary lists for graphics-related terminology (e.g., x/y positioning, radius, canvas, constant)

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Student restates information - Space for movement or breaks - Extra visual and verbal cues and prompts - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Pairing students with beginning English language skills with students who have more advanced English language skills

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Supplemental Instruction - Tiered Level of Supports - Tiered Activities - Small Group Instruction

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students

It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

-Homogenous Grouping - Small Group In-Depth Instruction - Differentiated Online Practice

Unit 4

Desired Results

Unit 4 Title: JavaScript Control Structures

Unit Summary:

In this unit, students expand their JavaScript skills by using control structures like if/else statements and loops to build interactive programs. They apply logic and comparison operators to create projects such as an inventory tracker, a digital jukebox, and a loop-based caterpillar drawing. In the following unit, students apply these concepts in more complex challenges, like a number guessing game, reinforcing problem-solving, logic, and user interaction.

Learning Standards

NJSLS:

- 8.1.12.DA.1 - Create interactive data visualizations using software tools to help others better understand real world phenomena, including climate change
- 8.1.12.DA.5 - Create data visualizations from large data sets to summarize, communicate, and support different interpretations of real-world phenomena
- 8.1.12.DA.6 - Create and refine computational models to better represent the relationships among different elements of data collected from a phenomenon or process
- 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
- 8.2.12.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

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CI.1 – Demonstrate the ability to reflect, analyze, and use creative skills and ideas
- 9.4.12.CI.2 – Identify career pathways that highlight personal talents, skills, and abilities
- 9.4.12.CT.1 – Identify problem-solving strategies used in the development of an innovative product or practice
- 9.4.12.CT.2 – Explain the potential benefits of collaborating to enhance critical thinking and problem solving
- 9.4.12.CT.4 – Participate in online strategy and planning sessions for course-based, school-based, or other project and determine the strategies that contribute to effective outcomes

- 9.4.12.DC.1 – Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content
- 9.4.12.DC.3 – Evaluate the social and economic implications of privacy in the context of safety, law, or ethics
- 9.4.12.DC.4 – Explain the privacy concerns related to the collection of data
- 9.4.12.GCA.1 – Collaborate with individuals to analyze a variety of potential solutions to climate change effects and determine why some solutions may work better than others
- 9.4.12.IML.3 – Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions
- 9.4.12.IML.4 – Assess and critique the appropriateness and impact of existing data visualizations for an intended audience
- 9.4.12.TL.1 – Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task
- 9.4.12.TL.3 – Analyze the effectiveness of the process and quality of collaborative environments
- 9.4.12.TL.4 – Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop and use a model based on evidence to illustrate the relationships between systems or between components of a system.
- Use a model to predict the relationships between systems or between components of a system.
- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Use mathematical representations of phenomena or design solutions to support and revise explanations.

- Apply scientific reasoning to link evidence to the claims to assess the extent to which the reasoning and data support the explanation or conclusion.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Construct an explanation based on valid and reliable evidence obtained from a variety of sources (including students' own investigations, models, theories, simulations, peer review) and the assumption that theories and laws that describe the natural world operate today as they did in the past and will continue to do so in the future.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How do control structures influence the flow of a program?
- When should you use a while loop versus a for loop?
- How can logical and comparison operators be used to make decisions in code?
- What strategies help break down complex problems using conditionals and loops?
- How can control structures be used to create interactive and user-responsive programs?

Unit Enduring Understandings:

- Control structures allow programs to make decisions and repeat actions, enabling dynamic and interactive behavior.
- If/else statements and loops are foundational tools for managing program flow and responding to conditions.
- Logical and comparison operators are essential for building meaningful conditions in decision-making.

	• Effective problem-solving in programming involves decomposing tasks and applying the right control structures. • Combining control structures with user input enhances the flexibility and usefulness of a program.
Vocabulary: booleans, if statement, else statement, else if statement, logical operators, comparison operators, conditionals, while loop, for loop, break statement, nested control structures, loop counter, infinite loop, iteration, control flow, user input, random number	
Students will be able to: • Create boolean variables to represent meaningful yes/no values • Print out the value of a boolean variable • Explain the purpose of if statements • Create their own if statements to selective choose which code is executed in their programs • Describe the meaning and usage of each logical operator: OR (||), AND (&&), and NOT (!) • Construct logical statements using boolean variables and logical operators • Explain the meaning of each of the comparison operators (<, <=, >, >=, ==, !=) • Create programs using the comparison operators to compare values • Predict the boolean result of comparing two values • Print out the boolean result of comparing values • Write graphics programs with conditionals • Use else if statements to check for multiple conditions • Explain the purpose of a while loop • Create while loops to repeat code while a condition is true • Utilize while loops to solve new types of problems • Explain how the loop-and-a-half structure is different from a traditional while loop	

- Explain what an infinite loop is
- Explain what the break statement does
- Write graphics programs that use while loops
- Use variables to update the position and size of graphics objects within a while loop
- Create for loops in JavaScript
- Explain the purpose of for loops
- Utilize for loops to avoid typing out repeated code
- Use the loop counter i inside the for loop code to do something different on each iteration
- Create graphics programs with for loops
- Use i to position graphics objects and change the size of graphics objects
- Compare and contrast while loops and for loops
- Define a problem in their own words and plan out a solution to the problem
- Break a large problem down into smaller pieces and solve each of the pieces, then use these solutions as building blocks to solve the larger problem
- Utilize the proper control structures to create general solutions
- Write clear and readable code using control structures, decomposition, and comments

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive coding labs focused on creating and using boolean variables, conditionals, and loops
- Guided exercises like the Inventory and Jukebox programs to connect real-world logic with control structures
- Visual programs using for loops (e.g., Caterpillar drawing) that integrate math, logic, and design
- Partner programming to practice decision-making and debug collaboratively
- Logic-based mini-games such as the Guessing Game to reinforce comparison operators and loop logic
- Journaling prompts where students reflect on decisions their programs make and how logic structures can be applied to real-life scenarios

Differentiation Strategies:

- For Advanced Learners: Introduce nested loops and more abstract logic puzzles (e.g., simulate a board game or a dynamic animation using control structures); encourage students to design original interactive tools or games involving multiple conditions and control flows
- For Struggling Learners: Use flowcharts and visual scaffolding to map out decision paths and loop structures; offer code templates with labeled blanks and guided questions to support step-by-step logic building
- For English Language Learners: Pair code concepts with real-life analogies and visual illustrations (e.g., traffic lights for conditionals); provide bilingual glossaries for control structure terms and encourage oral explanations of logic before coding
- Universal Design for Learning (UDL): Allow students to demonstrate understanding through diagrams, pseudocode, or verbal reasoning; use color-coded examples and manipulatives to illustrate how loops and conditionals function; provide multiple pacing options and digital tools that support code visualization

Education Connections:

- Examine decision-making systems in historical and social justice contexts (e.g., algorithms and ethical decision-making)
- Highlight the work of Asian American and Pacific Islander technologists in software development and logic-based systems
- Encourage the creation of projects that express advocacy, such as climate awareness or cultural representation through interactive programs
- Showcase how logic and control structures are used in accessibility tools, empowering people with disabilities through technology
- Promote student-designed games or apps that reflect diverse identities and inclusive narratives through decision trees and interactive paths

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Inventory

      Write a program that keeps track of a simple inventory for a store. While there are still items left in the inventory, ask the user how many items they would like to buy. Then print out how many are left in inventory after the purchase. You should use a while loop for this problem. Make sure you catch the case where the user tries to buy more items than there are in the inventory. In that case, you should print a message to the user saying that their request isn't possible.
    - Example Exercise: Caterpillar

      This graphics program should draw a caterpillar. A caterpillar has NUM_CIRCLES circles. Every other circle is a different color, the even circles are red, and the odd circles are green (by even we mean when i is an even number). Use a for loop to draw the caterpillar, centered vertically on the screen. Also, be sure that the caterpillar is still drawn across the whole canvas even if the value of NUM_CIRCLES is changed.
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Control Structures Challenges

Common Benchmark Assessments:

- Control Structures Challenges

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment with support for control structures and user input

- Interactive coding exercises focused on booleans, conditionals, loops, and logical operators
- Built-in debugging tools with real-time error analysis and code tracing
- JavaScript documentation and reference guides covering if/else statements, comparison and logical operators, while and for loops

Supplemental Materials:

- Visual logic organizers such as flowcharts, truth tables, and condition/loop planning templates
- Peer collaboration rubrics and pair programming strategies to support discussion and problem-solving
- Real-world coding scenarios and logic challenges to apply control structure concepts
- Assessment rubrics for evaluating correctness, problem-solving, use of control structures, and code clarity
- Culturally responsive project prompts that incorporate diverse perspectives in decision-based programming (e.g., inclusive guessing games or simulations)
- Anchor charts and vocabulary lists for control structure terms (e.g., boolean, condition, iteration, loop counter, infinite loop, break)

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Review of directions - Student restates information - Extra visual and verbal cues and prompts - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Simplified/Verbal Instructions - Visual learning, including graphic organizers - Teacher modeling - Scaffolding - Word walls - Think-pair-share - Cooperative learning groups

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Supplemental Instruction - Think-Pair-Share by Readiness - Differentiated Online Practice - Leveled Rubrics - Tiered Level of Supports - Tiered Activities - Small Group Instruction - Work Alone Choice - Group Work Choice

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students

It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

- Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments - Pre-Assess to provide Differentiated Instruction to Accelerate Learning

Unit 5

Desired Results

Unit 5 Title: JavaScript Functions

Unit Summary:

In this unit, students deepen their understanding of JavaScript by learning to write reusable code with functions, including the use of parameters, return values, default parameters, and variable scope. They practice breaking down

problems into modular pieces through exercises like creating an isEven() function to determine if numbers are even or odd. In the following challenges unit, students apply these skills to solve more complex problems, such as drawing colorful balloons with randomized attributes, reinforcing their ability to organize and simplify code using functions.

Learning Standards

NJSLS:

- 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
- 8.2.12.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

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CI.1 – Demonstrate the ability to reflect, analyze, and use creative skills and ideas
- 9.4.12.CI.2 – Identify career pathways that highlight personal talents, skills, and abilities
- 9.4.12.CT.1 – Identify problem-solving strategies used in the development of an innovative product or practice
- 9.4.12.CT.2 – Explain the potential benefits of collaborating to enhance critical thinking and problem solving
- 9.4.12.CT.4 – Participate in online strategy and planning sessions for course-based, school-based, or other project and determine the strategies that contribute to effective outcomes

- 9.4.12.DC.1 – Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content
- 9.4.12.DC.3 – Evaluate the social and economic implications of privacy in the context of safety, law, or ethics
- 9.4.12.DC.4 – Explain the privacy concerns related to the collection of data
- 9.4.12.GCA.1 – Collaborate with individuals to analyze a variety of potential solutions to climate change effects and determine why some solutions may work better than others
- 9.4.12.IML.3 – Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions
- 9.4.12.IML.4 – Assess and critique the appropriateness and impact of existing data visualizations for an intended audience
- 9.4.12.TL.1 – Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task
- 9.4.12.TL.3 – Analyze the effectiveness of the process and quality of collaborative environments
- 9.4.12.TL.4 – Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop and use a model based on evidence to illustrate the relationships between systems or between Develop a model based on evidence to illustrate the relationships between systems or between components of a system.
- Use a model to predict the relationships between systems or between components of a system.
- Apply scientific principles and evidence to provide an explanation of phenomena and solve design problems, taking into account possible unanticipated effects.
- Construct and revise an explanation based on valid and reliable evidence obtained from a variety of sources (including students' own investigations, models, theories, simulations, peer review) and the assumption that theories and laws that describe the natural world operate today as they did in the past and will continue to do so in the future.

- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Use mathematical representations of phenomena or design solutions to support and revise explanations.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Apply scientific reasoning to link evidence to the claims to assess the extent to which the reasoning and data support the explanation or conclusion.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How can functions be used to make code more organized, reusable, and easier to understand?
- What roles do parameters and return values play in defining and using functions?
- How does variable scope affect the behavior of functions and their interaction with the rest of a program?

Unit Enduring Understandings:

- Functions enable programmers to encapsulate code into reusable blocks, promoting modularity and reducing repetition.
- Parameters allow functions to accept input, while return values provide output, enabling dynamic and flexible behavior.

• In what ways can default parameters simplify function calls and improve flexibility? • How can complex problems be broken down and solved effectively using functions?	• Understanding variable scope is crucial for controlling data accessibility and preventing unintended side effects in programs. • Default parameters help manage optional inputs, making functions more versatile and easier to use. • Breaking down large problems into smaller functions improves problem-solving efficiency and code maintainability.
Vocabulary: functions, parameters, return values, default parameters, variable scope, boolean, reusability, function call, argument, local variable, global variable, encapsulation, debugging	
Students will be able to: • Explain the use of parameters and arguments • Create functions that take in parameters as input • Use parameters to generalize functions and reduce repeated code • Explain the purpose of returning a value from a function. • Create functions that return values. • Create programs that call functions with return values and store the result for later use. • Understand the role default values can have in a function. • Set default values for their parameters. • Properly set the order of parameters and the default values. • Identify the scope of a variable • Identify which variables are in scope at a given point in a program • Synthesize the skills and concepts from the The Canvas and Graphics, Control Structures, and the Functions modules to solve increasingly difficult programming challenges • Break down a large problem into smaller parts using Top Down Design, and solve each of these smaller parts using functions	

- Create helpful comments with preconditions and postconditions to help the reader understand the code
- Find and fix bugs in large programs

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive coding labs focused on writing and using functions with parameters, return values, and variable scope
- Guided exercises such as the isEven() program to practice defining and calling functions with Boolean return values
- Visual programming challenges like drawing balloons with random attributes to reinforce function usage and parameters
- Pair programming and collaborative debugging sessions to develop problem-solving and communication skills
- Top Down Design activities where students break down larger problems into smaller, function-based parts
- Reflective journaling prompts encouraging students to explain how functions improve code organization and reuse

Differentiation Strategies:

- For Advanced Learners: Introduce recursive functions and explore more complex parameter passing (e.g., multiple default parameters); challenge students to design original projects using higher-order functions or function callbacks
- For Struggling Learners: Use flowcharts and step-by-step pseudocode to visualize function inputs and outputs; provide partially completed code templates with guiding questions to scaffold function creation
- For English Language Learners: Pair programming with visual analogies (e.g., “functions as machines that take inputs and return outputs”); bilingual glossaries for function-related vocabulary; encourage verbal explanation of code logic before coding
- Universal Design for Learning (UDL): Offer multiple ways to demonstrate understanding—writing code, creating diagrams of function flow, or explaining concepts verbally; use color-coded examples to highlight parameters,

return statements, and scope; allow flexible pacing and provide digital tools to visualize function execution

Education Connections:

- Explore ethical decision-making and problem-solving through functions in social justice contexts (e.g., algorithms in fair resource allocation)
- Highlight contributions of Asian American and Pacific Islander computer scientists in software design and modular programming
- Encourage projects that raise awareness on climate change or social issues using functions for simulation or data visualization
- Showcase how software with well-designed functions supports accessibility tools, empowering people with disabilities
- Promote creation of inclusive programs that reflect diverse narratives through customizable function-driven interactive stories or games

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Is it even?

      Write a program that continually asks the user for integers and the prints whether their input is even or odd. The user should keep entering numbers until they enter 0; at that point, print "Done!" on its own line. In order to check if the inputted integer is even or odd, you should define and call a function named `isEven()`. This function should return a Boolean value of `true` or `false` depending if the number is even or not.
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Functions Challenges

Common Benchmark Assessments:

- Functions Challenges

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment with full support for defining and calling functions, parameters, return values, and variable scope
- Interactive coding exercises focused on writing functions with parameters, using return values, and managing variable scope
- Built-in debugging tools with real-time error analysis and function call tracing
- JavaScript documentation and reference guides covering function syntax, default parameters, scope rules, and best practices for reusable code

Supplemental Materials:

- Visual organizers such as function flow diagrams, parameter mapping charts, and Top Down Design planning templates
- Peer collaboration rubrics and pair programming strategies to support discussion, code review, and debugging of functions
- Real-world coding scenarios and function-based challenges to apply modular programming concepts (e.g., isEven(), balloon drawing)
- Assessment rubrics for evaluating correctness, problem decomposition, effective use of functions, and code clarity
- Culturally responsive project prompts that incorporate diverse perspectives through function-driven interactive programs (e.g., inclusive animations, social issue simulations)

- Anchor charts and vocabulary lists for function-related terms (e.g., parameter, return value, scope, default parameter, argument, encapsulation)

Accommodations and Modifications:
Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Review of directions - Student restates information - Extra visual and verbal cues and prompts - Preferential seating - Follow a routine/schedule - Rest breaks - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Simplified/Verbal Instructions - Frequent Breaks - Visual learning, including graphic organizers - Teacher modeling - Pairing students with beginning English language skills with students who have more advanced English language skills - Scaffolding - Word walls - Think-pair-share - Cooperative learning groups

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Learning Contracts - Supplemental Instruction - Homework Options - Differentiated Online Practice - Leveled Rubrics - Tiered Level of Supports - Tiered Activities - Small Group Instruction - Work Alone Choice - Group Work Choice

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students
It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

- Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments - Pre-Assess to provide Differentiated Instruction to Accelerate Learning

Unit 6

Desired Results

Unit 6 Title: Animations

Unit Summary:

In this unit, students learn to create animations and interactive programs using JavaScript, including timers, randomization, and mouse and keyboard events. They practice with exercises like paint splatters and moving targets, then apply these skills in animation challenges. Finally, students build two games—Breakout and Snake—reinforcing concepts like graphics, collision detection, and user input handling.

Learning Standards

NJSLS:

- 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
- 8.2.12.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

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CI.1 – Demonstrate the ability to reflect, analyze, and use creative skills and ideas
- 9.4.12.CI.2 – Identify career pathways that highlight personal talents, skills, and abilities
- 9.4.12.CT.1 – Identify problem-solving strategies used in the development of an innovative product or practice
- 9.4.12.CT.2 – Explain the potential benefits of collaborating to enhance critical thinking and problem solving
- 9.4.12.CT.4 – Participate in online strategy and planning sessions for course-based, school-based, or other project and determine the strategies that contribute to effective outcomes
- 9.4.12.DC.1 – Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content
- 9.4.12.DC.3 – Evaluate the social and economic implications of privacy in the context of safety, law, or ethics
- 9.4.12.DC.4 – Explain the privacy concerns related to the collection of data
- 9.4.12.GCA.1 – Collaborate with individuals to analyze a variety of potential solutions to climate change effects and determine why some solutions may work better than others
- 9.4.12.IML.3 – Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions
- 9.4.12.IML.4 – Assess and critique the appropriateness and impact of existing data visualizations for an intended audience

- 9.4.12.TL.1 – Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task
- 9.4.12.TL.3 – Analyze the effectiveness of the process and quality of collaborative environments
- 9.4.12.TL.4 – Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop and use a model based on evidence to illustrate the relationships between systems or between components of a system.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Apply scientific ideas to solve a design problem, taking into account possible unanticipated effects.
- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Construct an explanation based on valid and reliable evidence obtained from a variety of sources (including students' own investigations, models, theories, simulations, peer review) and the assumption that theories and laws that describe the natural world operate today as they did in the past and will continue to do so in the future.
- Design or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.

- HS-ETS1-3 Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How can timers and randomization be used to create dynamic animations in JavaScript?
- In what ways do mouse and keyboard events enable user interaction in games?
- How does collision detection contribute to gameplay mechanics?
- What strategies can be used to break down complex animations or games into manageable code components?
- How can reusable functions improve the organization and scalability of animation and game code?

Unit Enduring Understandings:

- Timers and randomization allow programs to produce animations that change over time and create unpredictable behaviors.
- Mouse and keyboard events are fundamental for enabling real-time user interaction in animations and games.
- Collision detection is essential for implementing game rules and interactions between objects.
- Breaking down animations and games into smaller parts makes programming more manageable and helps debug and extend code.
- Using reusable functions enhances code clarity, reduces repetition, and simplifies the development of complex animations and games.

Vocabulary: timers, randomization, mouse events, keyboard events, animation, collision detection, graphics, user input, event-driven programming, functions, variables, loops, conditionals, sprites, game mechanics, canvas, debugging, interactive programs, code modularity, scalable code

Students will be able to:

- Explain in their own words how animation works
- Create animation in programs using the setTimer function
- Explain what a callback function is
- Create programs with timers to create increasingly challenging animations
- Stop animation timers when a condition is met using stopTimer() function
- Understand when objects “collide” with the canvas walls and other objects.
- Write their own collision detection logic.
- Describe how events are different than timers
- Use mouse click events to create programs that respond to user clicks
- Explain how events are different from timers.
- Create interactive programs that use events to respond to the mouse moving
- Explain how events are different from timers.
- Create interactive programs that use events to respond to the keyboard input.
- Define a problem in their own words and plan out a solution to the problem
- Break a large problem down into smaller pieces and solve each of the pieces, then use these solutions as building blocks to solve the larger problem
- Utilize the proper control structures to create general solutions
- Write clear and readable code using timers, events, control structures, functions, decomposition, and comments
- Synthesize the skills and concepts from Control Structures, Functions, and Animation and Games to create their very own Breakout game from scratch!
- Break down a large problem into smaller parts using Top Down Design, and solve each of these smaller parts using functions
- Create helpful comments with preconditions and postconditions to help the reader understand the code
- Find and fix bugs in large programs

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive coding labs focused on using timers, events, and collision detection to create animations and interactive games
- Guided exercises such as creating paint splatter effects, moving targets, and a basic snake game to practice event handling and timers
- Animation challenges that require students to build responsive, interactive programs with mouse and keyboard inputs
- Pair programming and collaborative debugging sessions to develop problem-solving, communication, and event-driven programming skills
- Top Down Design activities where students break down game mechanics into smaller functions and event handlers
- Reflective journaling prompts encouraging students to explain how timers, events, and collision logic contribute to interactive program behavior

Differentiation Strategies:

- For Advanced Learners: Explore more complex animations using simultaneous timers and multiple event listeners; challenge students to create original games incorporating advanced collision logic or custom event triggers
- For Struggling Learners: Use visual aids like flowcharts and pseudocode to map out timer sequences and event flows; provide scaffolded templates with clear instructions for adding timers and event handlers
- For English Language Learners: Pair programming with visual analogies (e.g., timers as clocks triggering actions, events as “listeners” waiting for input); bilingual glossaries for animation and event vocabulary; encourage verbal explanation of animation steps before coding
- Universal Design for Learning (UDL): Provide multiple modes of engagement—coding animations, creating storyboards for interactive games, or verbally describing event responses; use color-coded code examples for timers, events, and collision logic; allow flexible pacing and interactive debugging tools to visualize animations in

real time

Education Connections:

- Examine ethical and social justice issues through game scenarios, such as simulating resource distribution or historical events using animations
- Highlight contributions of Asian American and Pacific Islander computer scientists in animation and interactive media fields
- Encourage projects that use animation to raise awareness about climate change, such as visualizing environmental data through moving graphics
- Demonstrate how accessible interactive games can empower people with disabilities by incorporating inclusive design and input methods
- Promote creation of games and animations that reflect diverse stories and perspectives, fostering empathy and cultural awareness through interactive media

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
    - Example Exercise: Paint Splatter

      Write a program that splatters paint on the screen every DELAY milliseconds. To splatter paint, pick a random color and draw CIRCLES_PER_SPLATTER circles of that color at random places on the screen. The radius of each circle should be a random value between MIN_RADIUS and MAX_RADIUS. Remember to use helper functions.
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Animations Challenges

Common Benchmark Assessments:

- Animations Challenges

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment fully supporting timers, event handling (mouse and keyboard), collision detection, and animation techniques
- Interactive coding exercises focused on using timers, creating callback functions, handling mouse and keyboard events, and writing collision detection logic
- Built-in debugging tools with real-time error analysis, event tracing, and animation step visualization
- JavaScript documentation and reference guides covering timer functions (setTimer, stopTimer), event listeners, collision detection algorithms, and best practices for interactive programming

Supplemental Materials:

- Visual organizers such as event flow diagrams, timer sequence charts, collision detection logic maps, and Top Down Design planning templates for breaking down game features
- Peer collaboration rubrics and pair programming strategies to support discussion, code review, and debugging of animations and interactive event-driven programs
- Real-world coding scenarios and animation/game challenges to apply concepts like event-driven interaction, timers, and collision detection (e.g., paint splatter, moving targets, Breakout game)
- Assessment rubrics for evaluating animation smoothness, responsiveness to user input, accuracy of collision detection, problem decomposition, and code clarity
- Culturally responsive project prompts that incorporate diverse perspectives through interactive animations and games addressing social issues, inclusivity, and environmental themes

- Anchor charts and vocabulary lists for animation and event-driven programming terms (e.g., timer, callback function, event listener, collision detection, mouse event, keyboard event, control structures)

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:

Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Review of directions - Extra visual and verbal cues and prompts - Preferential seating - Follow a routine/schedule - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners

English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Simplified/Verbal Instructions - Visual learning, including graphic organizers - Teacher modeling - Pairing students with beginning English language skills with students who have more advanced English language skills - Scaffolding - Word walls

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Learning Contracts - Supplemental Instruction - Think-Pair-Share by Readiness - Leveled Rubrics - Tiered Level of Supports - Small Group Instruction - Group Work Choice

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.

504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students
It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

- Small Group In-Depth Instruction - Alternative Assessments - Pre-Assess to provide Differentiated Instruction to Accelerate Learning

Unit 7

Desired Results

Unit 7 Title: Data Structures

Unit Summary:

In this unit, students explore key data structures in JavaScript, beginning with arrays—learning to create, iterate, and manipulate them using built-in methods. They then dive into objects, developing properties, methods, constructors, and techniques for iterating through object data. Applying these concepts, students build a classic Tic-Tac-Toe game that combines data structures with graphical elements. Finally, they demonstrate their programming skills by creating the Helicopter game, incorporating graphics, collision detection, scrolling backgrounds, and random obstacle generation.

Learning Standards

NJSLS:

- 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
- 8.2.12.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

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CI.1 – Demonstrate the ability to reflect, analyze, and use creative skills and ideas
- 9.4.12.CI.2 – Identify career pathways that highlight personal talents, skills, and abilities
- 9.4.12.CT.1 – Identify problem-solving strategies used in the development of an innovative product or practice
- 9.4.12.CT.2 – Explain the potential benefits of collaborating to enhance critical thinking and problem solving
- 9.4.12.CT.4 – Participate in online strategy and planning sessions for course-based, school-based, or other project and determine the strategies that contribute to effective outcomes
- 9.4.12.DC.1 – Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content
- 9.4.12.DC.3 – Evaluate the social and economic implications of privacy in the context of safety, law, or ethics
- 9.4.12.DC.4 – Explain the privacy concerns related to the collection of data
- 9.4.12.GCA.1 – Collaborate with individuals to analyze a variety of potential solutions to climate change effects and determine why some solutions may work better than others
- 9.4.12.IML.3 – Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions
- 9.4.12.IML.4 – Assess and critique the appropriateness and impact of existing data visualizations for an intended audience
- 9.4.12.TL.1 – Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task

- 9.4.12.TL.3 – Analyze the effectiveness of the process and quality of collaborative environments
- 9.4.12.TL.4 – Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop a model based on evidence to illustrate the relationships between systems or between components of a system.
- Use a model to predict the relationships between systems or between components of a system.
- Analyze data using tools, technologies, and/or models (e.g., computational, mathematical) in order to make valid and reliable scientific claims or determine an optimal design solution.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.
- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Use mathematical and/or computational representations of phenomena or design solutions to support explanations.
- Design or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Evaluate a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-3 Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.

- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How can arrays and objects be used to organize and manage data effectively in programming?
- What are the advantages of using built-in methods to manipulate arrays and objects?
- How do constructors help in creating reusable and scalable objects?
- In what ways can data structures be combined with graphics to create interactive applications and games?
- How do programming concepts like collision detection and randomization enhance gameplay experience in game development?

Unit Enduring Understandings:

- Arrays and objects are fundamental data structures that enable efficient storage, access, and manipulation of related data in programs.
- Built-in methods simplify working with arrays and objects, making code more readable and maintainable.
- Constructors provide a blueprint for creating multiple instances of objects, supporting modular and scalable code design.
- Combining data structures with graphical elements allows developers to build interactive and dynamic applications.
- Game development relies on concepts such as collision detection, scrolling backgrounds, and random events to create engaging user experiences.

Vocabulary: arrays, array methods, iteration, objects, object properties, data structures, collision detection, indexing

Students will be able to:

- Define an array
- Access elements of an array with an index
- Add elements at the end of an array using the push method

- Remove elements from the end of an array using the pop method
- Determine the length of an array using the length property
- Use the length of an array and a for loop to loop through the elements in an array
- Use a for...of loop to access each element in the array without using their indices.
- Explain why arrays and array iteration are very useful when creating graphics programs with many graphics elements.
- Use array iteration to control the graphics elements in a program.
- Understand what objects are and how they store data.
- Understand the difference between an object's key and value.
- Create an empty object and assign it a key/value pair.
- Define and manipulate objects that store graphic properties.
- Pass objects as parameters and explain the difference between passing by value and passing by reference.
- Use objects to create and manage animated elements on a canvas.
- Define and implement methods inside JavaScript objects.
- Differentiate between object properties and methods.
- Use object methods to manipulate and interact with objects.
- Iterate through the keys of an object
- Better understand the difference between keys and values in an object
- Define and implement an object constructor in JavaScript.
- Differentiate between object literals and object constructors.
- Use constructors to create multiple instances of an object with shared properties and methods.

Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])*

Learning Activities:

- Interactive labs introducing arrays and objects through hands-on exercises (e.g., creating travel lists, shopping carts, and digital artwork objects)

- Scaffolded coding challenges focused on array iteration, object methods, and constructors
- Guided programming projects including Tic Tac Toe and Helicopter Game, integrating data structures with animation, graphics, and user input
- Mini design sprints where students plan out object models and use pseudocode to design constructors and object-based animations
- Pair programming sessions to collaboratively build and debug programs using arrays and objects
- Reflection activities where students explain the difference between object literals and constructors, or the impact of using objects in game design

Differentiation Strategies:

- For Advanced Learners: Challenge students to design their own reusable object constructors for game entities; extend the Helicopter game with additional object types (e.g., power-ups or enemies) and complex array manipulations
- For Struggling Learners: Provide visual diagrams of arrays and objects; use code scaffolds with step-by-step tasks; offer example-based mini-lessons on push, pop, and object properties
- For English Language Learners: Include bilingual coding vocabulary lists (e.g., array = lista, object = objeto); encourage pair programming with discussion prompts; use icons or diagrams to represent key/value pairs and array indices
- Universal Design for Learning (UDL): Offer multiple ways to demonstrate understanding—coding, drawing object models, or narrating the function of a constructor; use color-coded examples for syntax highlighting and key concepts; support real-time debugging with visual output and error explanations

Education Connections:

- Design games like Tic Tac Toe or Helicopter that incorporate themes of fairness, justice, or access to resources, linking to Amistad or social equity
- Explore Asian American and Pacific Islander contributions to software engineering and data visualization
- Create visual simulations (e.g., barcodes or environmental data sets) to represent climate change impacts using

arrays and graphics
- Promote accessibility by modifying the Helicopter game for alternative input methods, highlighting how code structure (objects and arrays) supports inclusive design
- Encourage storytelling through object-oriented animation projects that reflect diverse perspectives, such as hobbies, family structures, or identities from underrepresented communities

Assessment Evidence

Formative Assessments:

- CodeHS coding exercises
  - Example Exercise: Bank Account

    In this exercise, you need to define a function `createAccount()` that creates a new bank account object and returns it.
  - Example Exercise: Starry Night

    In this program, you have been given an `artwork` object with properties and a method. You need to complete the two functions `printAllKeys()` and `printProperties()` so that the former prints every key in the parameter object and the latter p
- CodeHS debugging exercises
- CodeHS Checks for Understanding

Summative Assessments:

- Project: Tic Tac Toe
- Project: Helicopter Game

Common Benchmark Assessments:

- Project: Tic Tac Toe
- Project: Helicopter Game

Alternative Assessments:

- CodeHS Unit Quiz

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment fully supporting arrays, objects, constructors, and graphical rendering for game development
- Interactive coding exercises focused on array manipulation, object creation, property access, method definition, and iteration through data structures
- Built-in debugging tools with real-time output, error feedback, and object/array visualizers to help students trace and troubleshoot data structure behavior
- JavaScript documentation and reference guides covering array methods (e.g., push, pop, length), object syntax, key/value access, constructors, and best practices for organizing data
- Structured project templates for Tic Tac Toe and Helicopter game, scaffolded to help students implement and extend game logic using data structures

Supplemental Materials:

- Visual organizers such as array mapping charts, object structure diagrams, constructor blueprints, and data flow visuals to support comprehension and planning
- Peer collaboration rubrics and pair programming strategies to support discussion, debugging, and shared understanding of array/object manipulation in games
- Real-world project challenges to apply concepts like using arrays for storing player moves, using objects for defining game entities, and constructors for reusable components
- Assessment rubrics for evaluating effective use of arrays and objects, clarity and reuse of code, functionality of constructors, and integration of data with graphics
- Culturally responsive project prompts encouraging students to create games or data-driven animations that reflect diverse experiences, raise awareness of social issues, or represent underrepresented communities

- Anchor charts and vocabulary lists for data structures and object-oriented programming terms (e.g., array, index, push, pop, length, object, key, value, method, constructor, instance, property)

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:
Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Review of directions - Student restates information - Extra visual and verbal cues and prompts - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners
English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Simplified/Verbal Instructions - Visual learning, including graphic organizers - Teacher modeling - Scaffolding

Students at Risk of Failure

- Alternative Assessments - Think-Pair-Share by Readiness - Differentiated Online Practice - Tiered Activities - Small Group Instruction - Group Work Choice

Students with 504 Plans
Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students
It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.
- Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments

Unit 8

Desired Results

Unit 8 Title: Creating JavaScript Games

Unit Summary:

In this final unit, students synthesize all the skills developed throughout the course to design and build an original game from scratch. They apply concepts such as basic graphics, collision detection, scrolling backgrounds, and random obstacle generation to bring their creative vision to life. Over the course of four weeks, students plan, code, and refine their projects, demonstrating their mastery of programming fundamentals and game design through a fully functional, self-directed game.

Learning Standards

NJSLS:

- 8.1.12.CS.1: Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
- 8.1.12.CS.4: Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
- 8.1.12.IC.1: Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
- 8.1.12.IC.2: Test and refine computational artifacts to reduce bias and equity deficits.
- 8.1.12.IC.3: Predict the potential impacts and implications of emerging technologies on larger social, economic, and political structures, using evidence from credible sources.
- 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.
- 8.2.12.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.
- 8.2.12.ITH.1: Analyze a product to determine the impact that economic, political, social, and/or cultural factors have had on its design, including its design constraints.
- 8.2.12.EC.1: Analyze controversial technological issues and determine the degree to which individuals, businesses, and governments have an ethical role in decisions that are made.
- 8.2.12.EC.3: Synthesize data, analyze trends, and draw conclusions regarding the effect of a technology on the individual, culture, society, and environment and share this information with the appropriate audience.

Interdisciplinary Connections

Career Readiness, Life Literacies, & Key Skills *(CLKS)*:

- 9.4.12.CI.1: Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
- 9.4.12.CI.2: Identify career pathways that highlight personal talents, skills, and abilities (e.g., 1.4.12prof.CR2b, 2.2.12.LF.8).
- 9.4.12.CI.3: Investigate new challenges and opportunities for personal growth, advancement, and transition (e.g., 2.1.12.PGD.1).
- 9.4.12.CT.1: Identify problem-solving strategies used in the development of an innovative product or practice (e.g., 1.1.12acc.C1b, 2.2.12.PF.3).
- 9.4.12.CT.2: Explain the potential benefits of collaborating to enhance critical thinking and problem solving (e.g., 1.3E.12profCR3.a).
- 9.4.12.CT.3: Enlist input from a variety of stakeholders (e.g., community members, experts in the field) to design a service learning activity that addresses a local or global issue (e.g., environmental justice).

- 9.4.12.CT.4: Participate in online strategy and planning sessions for course-based, school-based, or other project and determine the strategies that contribute to effective outcomes.
- 9.4.12.DC.1: Explain the beneficial and harmful effects that intellectual property laws can have on the creation and sharing of content (e.g., 6.1.12.CivicsPR.16.a).
- 9.4.12.DC.3: Evaluate the social and economic implications of privacy in the context of safety, law, or ethics (e.g., 6.3.12.HistoryCA.1).
- 9.4.12.DC.4: Explain the privacy concerns related to the collection of data (e.g., cookies) and generation of data through automated processes that may not be evident to users (e.g., 8.1.12.NI.3).
- 9.4.12.DC.5: Debate laws and regulations that impact the development and use of software.
- 9.4.12.DC.6: Select information to post online that positively impacts personal image and future college and career opportunities.
- 9.4.12.DC.7: Evaluate the influence of digital communities on the nature, content and responsibilities of careers, and other aspects of society (e.g., 6.1.12.CivicsPD.16.a).
- 9.4.12.DC.8: Explain how increased network connectivity and computing capabilities of everyday objects allow for innovative technological approaches to climate protection.
- 9.4.12.GCA.1: Collaborate with individuals to analyze a variety of potential solutions to climate change effects and determine why some solutions (e.g., political, economic, cultural) may work better than others (e.g., SL.11-12.1., HS-ETS1-1, HS-ETS1-2, HS-ETS1-4, 6.3.12.GeoGI.1, 7.1.IH.IPERS.6, 7.1.IL.IPERS.7, 8.2.12.ETW.3).
- 9.4.12.IML.2: Evaluate digital sources for timeliness, accuracy, perspective, credibility of the source, and relevance of information, in media, data, or other resources (e.g., NJSLSA.W8, Social Studies Practice: Gathering and Evaluating Sources).
- 9.4.12.IML.3: Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions (e.g., S-ID.B.6a., 8.1.12.DA.5, 7.1.IH.IPRET.8).
- 9.4.12.IML.4: Assess and critique the appropriateness and impact of existing data visualizations for an intended audience (e.g., S-ID.B.6b, HS-LS2-4).
- 9.4.12.IML.5: Evaluate, synthesize, and apply information on climate change from various sources appropriately (e.g., 2.1.12.CHSS.6, S.IC.B.4, S.IC.B.6, 8.1.12.DA.1, 6.1.12.GeoHE.14.a, 7.1.AL.PRSNT.2).

- 9.4.12.IML.6: Use various types of media to produce and store information on climate change for different purposes and audiences with sensitivity to cultural, gender, and age diversity (e.g., NJSLSA.SL5).
- 9.4.12.IML.7: Develop an argument to support a claim regarding a current workplace or societal/ethical issue such as climate change (e.g., NJSLSA.W1, 7.1.AL.PRSNT.4).
- 9.4.12.IML.8: Evaluate media sources for point of view, bias, and motivations (e.g., NJSLSA.R6, 7.1.AL.IPRET.6).
- 9.4.12.IML.9: Analyze the decisions creators make to reveal explicit and implicit messages within information and media (e.g., 1.5.12acc.C2a, 7.1.IL.IPRET.4).
- 9.4.12.TL.1: Assess digital tools based on features such as accessibility options, capacities, and utility for accomplishing a specified task (e.g., W.11-12.6.).
- 9.4.12.TL.3: Analyze the effectiveness of the process and quality of collaborative environments.
- 9.4.12.TL.4: Collaborate in online learning communities or social networks or virtual worlds to analyze and propose a resolution to a real-world problem (e.g., 7.1.AL.IPERS.6).
- 9.2.12.CAP.2: Develop college and career readiness skills by participating in opportunities such as structured learning experiences, apprenticeships, and dual enrollment programs.
- 9.2.12.CAP.4: Evaluate different careers and develop various plans (e.g., costs of public, private, training schools) and timetables for achieving them, including educational/training requirements, costs, loans, and debt repayment.
- 9.2.12.CAP.6: Identify transferable skills in career choices and design alternative career plans based on those skills.
- 9.2.12.CAP.14: Analyze and critique various sources of income and available resources (e.g., financial assets, property, and transfer payments) and how they may substitute for earned income.

Interdisciplinary Connections:

Science and Engineering Practices:

- Develop and use a model based on evidence to illustrate the relationships between systems or between components of a system.
- Create a computational model or simulation of a phenomenon, designed device, process, or system.

- Design, evaluate, and/or refine a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Evaluate a solution to a complex real-world problem, based on scientific knowledge, student-generated sources of evidence, prioritized criteria, and tradeoff considerations.
- Apply scientific reasoning to link evidence to the claims to assess the extent to which the reasoning and data support the explanation or conclusion.

Engineering Design Standards:

- HS-ETS1-2 Design a solution to a complex real-world problem by breaking it down into smaller, more manageable problems that can be solved through engineering.
- HS-ETS1-3 Evaluate a solution to a complex real-world problem based on prioritized criteria and trade-offs that account for a range of constraints, including cost, safety, reliability, and aesthetics, as well as possible social, cultural, and environmental impacts.
- HS-ETS1-4 Use a computer simulation to model the impact of proposed solutions to a complex real-world problem with numerous criteria and constraints on interactions within and between systems relevant to the problem.

Learning Targets

Unit Essential Questions:

- How can I combine programming concepts to design a fully functional and engaging game?
- What makes a game both playable and enjoyable for users?
- How do planning and iteration contribute to successful game development?
- In what ways can graphics and interactivity enhance the player experience?

Unit Enduring Understandings:

- Successful game development requires the integration of multiple programming skills, including graphics, logic, and user interaction.
- Game design involves balancing challenge and fun while maintaining clarity and consistency in gameplay.
- Planning, prototyping, and iterating are critical to transforming an idea into a polished final product.

• How can I troubleshoot and solve problems independently during the game development process?	• Visual and interactive elements significantly impact user engagement and game quality. • Debugging and problem-solving are essential skills for independent, real-world programming tasks.
Vocabulary: basic graphics, collision detection, scrolling background, random obstacles, game loop, sprite, hitbox, event listener, frame rate, animation, function, variable, array, object, constructor, loop, conditionals, debugging, user input, key press, mouse click, logic, pseudocode, prototype, iteration, design plan, playtesting, feedback, physics, gravity, velocity	
Students will be able to: • Demonstrate the ability to integrate multiple programming concepts and skills learned throughout the course into a cohesive final project. • Plan and design a game project that applies core elements such as graphics, collision detection, and game mechanics. • Evaluate and prioritize game features to define a clear project scope that is achievable within the given timeframe. • Modify and simplify the project plan by eliminating or adjusting features to ensure timely completion. • Develop and implement the game according to the scoped design, applying best practices for coding and debugging. • Prepare a presentation that explains the game's concept, development process, and key programming techniques used. • Communicate effectively with peers by sharing the project and responding to questions or feedback.	
Learning Activities & Differentiation: *(Amistad & Holocaust, Asian American & Pacific Islanders, Climate Change, LGBTQ and People with Disabilities, Diversity, Equity & Inclusion [DEI])* **Learning Activities:** • Guided project planning sessions where students brainstorm and outline game concepts emphasizing core	

programming elements such as graphics, collision detection, and game mechanics.
- Scaffolded coding workshops to develop key features step-by-step, including implementing scrolling backgrounds, random obstacle generation, and collision logic.
- Peer review and iterative design cycles where students test each other's games and provide constructive feedback for improvement.
- Collaborative pair programming sessions focused on debugging, feature implementation, and scope management to ensure projects remain achievable within the timeframe.
- Presentation rehearsals where students prepare to explain their game's design, programming techniques, and development challenges to classmates.
- Reflective journaling or discussion prompts encouraging students to connect their game themes to social justice, cultural representation, or accessibility.

Differentiation Strategies:

- For Advanced Learners: Encourage adding sophisticated features such as adaptive difficulty, multiple levels, or additional object types; support the use of advanced programming concepts like state machines or event handling to enrich gameplay.
- For Struggling Learners: Provide step-by-step guides, pre-built code templates, and visual aids (flowcharts, diagrams) to support coding complex features; offer one-on-one check-ins to scaffold problem-solving.
- For English Language Learners: Supply bilingual glossaries of key game development and programming terms; use visual coding blocks or icons to illustrate concepts; encourage collaborative work with peers for language and coding support.
- Universal Design for Learning (UDL): Offer flexible ways to demonstrate learning such as video walkthroughs, annotated screenshots, or live coding demonstrations; use color-coded code examples and real-time visual debugging tools; provide options to modify game controls to accommodate diverse physical or cognitive needs.

Education Connections:

- Integrate themes of fairness and equity by designing game mechanics that simulate social justice concepts,

connecting to Amistad and Holocaust studies.
- Highlight contributions from Asian American and Pacific Islander technologists by exploring culturally relevant game themes or stories.
- Use game elements to visualize environmental data or climate change impacts, fostering awareness through interactive design.
- Promote accessibility by designing games that accommodate alternative input methods or visual/audio aids, linking programming structure to inclusive technology design.
- Encourage storytelling through game narratives or character design that reflect LGBTQ+ identities, disabilities, or other diverse experiences, supporting student expression and cultural representation.

Assessment Evidence

Formative Assessments:

- CodeHS exercises
    - Example Exercise: Planning and Design

      For this exercise, you should each list the top two or three ideas you want to build. After coming up with ideas on your own, share the ideas with one another and discuss their viability. You may want to also discuss your ideas with your teacher to make sure your team is taking on a challenge, but not overreaching.

      For each of your ideas, state the following information:

        - the title of your project
        - the goal of your project (What does it do? Pretend you are giving a summary to sell your idea.)
        - how the project will meet the requirements. What components will you use and what will their purpose be?
        - how the user will interact with the program. Which piece(s) of the program will they press and what will the effect be?
- CodeHS debugging exercises

- CodeHS Checks for Understanding

Summative Assessments:

- Final Project: Create Your Own Game

Common Benchmark Assessments:

- Final Project: Create Your Own Game

Alternative Assessments:

- MC Assessment

Core Instructional & Supplemental Materials

Core Materials:

- JavaScript coding environment fully supporting game development features such as graphics rendering, collision detection, scrolling backgrounds, and random obstacle generation.
- Interactive coding exercises focused on integrating graphics, implementing collision logic, managing game states, and using functions and variables to control gameplay.
- Built-in debugging tools with real-time visual output, error feedback, and performance monitoring to help students trace game logic, fix bugs, and optimize code.
- JavaScript documentation and reference guides covering key game development concepts such as event handling, animation loops, collision detection algorithms, randomization techniques, and best practices for modular game design.
- Structured project planning templates and coding scaffolds to guide students in designing, building, and iterating on their original games within the project scope.

Supplemental Materials:

- Visual organizers including game flowcharts, collision detection diagrams, sprite maps, and randomization flow visuals to support game design and debugging.
- Peer collaboration rubrics and pair programming strategies to foster teamwork, problem-solving, and shared code ownership during game development.

- Real-world project challenges encouraging application of learned skills to create games that integrate graphics, interaction, and game mechanics effectively.
- Assessment rubrics focused on game functionality, code quality, creativity, implementation of core programming concepts, and effective project scope management.
- Culturally responsive project prompts inviting students to develop games that explore themes related to social justice, environmental issues, diversity, and inclusion.
- Anchor charts and vocabulary lists highlighting game development and programming terminology such as collision detection, sprite, game loop, event listener, velocity, debugging, scope, iteration, and randomization.

Accommodations and Modifications: Students with Disabilities, English Language Learners, Students at Risk of Failure, Students with 504s, Gifted & Talented Students

Specific Strategies and Practices that Support Students with Disabilities:
Students receiving Special Education programming have specific goals and objectives, as well as accommodations and modifications outlined within their Individualized Education Plans (IEP) due to an identified disability and/or diagnosis. In addition to exposure to the general education curriculum, the instruction is differentiated based upon the student's needs. The IEP acts as a supplemental curriculum guide inclusive of instructional strategies that support each learner.

- Small group/One to one - Additional time - Review of directions - Student restates information - Extra visual and verbal cues and prompts - Preferential seating - Follow a routine/schedule - Verbal and visual cues regarding directions and staying on task - Checklists - Immediate feedback

Specific Strategies and Practices that Support English Language Learners
English language learner (ELL) is a national-origin-minority student who is limited-English-proficient. This term is often preferred over limited-English-proficient (LEP) as it highlights accomplishments rather than deficits.

- Pre-teaching of vocabulary and concepts - Personal Glossary - Extended Time - Simplified/Verbal Instructions - Visual learning, including graphic organizers - Teacher modeling - Pairing students with beginning English language skills with students who have more advanced English language skills - Scaffolding - Word walls - Think-pair-share

Students at Risk of Failure

- Home/School Connection - Alternative Assessments - Supplemental Instruction - Homework Options - Differentiated Online Practice - Leveled Rubrics - Small Group Instruction - Work Alone Choice - Group Work Choice

Students with 504 Plans

Teachers are responsible for implementing designated services and strategies identified on a student's 504 Plan.
504 plans are formal plans that schools develop to give kids the support they need and covers any condition that limits daily activities in a major way. Students' accommodations are outlined in their individual 504 plan.

Specific Strategies and Practices that Support Gifted & Talented Students

It is critical that all teachers are able to recognize a high-ability student who may need more depth and complexity in instruction.

-Small Group In-Depth Instruction - Differentiated Online Practice - Alternative Assessments - Pre-Assess to provide Differentiated Instruction to Accelerate Learning