JCDS
BOSTON'S JEWISH COMMUNITY DAY SCHOOL
בית ספר יהודי קהילתי

JCDS *Coding as Literacy* Framework

In a world where driverless cars are no longer far-fetched science fiction, news feeds are formed by marketing algorithms, unlimited goods and services are bought and sold online and cyber-warfare is a looming national security threat, computer science competency is a vital skill for full participation in 21st century civic society. Coding engages children in problem solving, abstract thinking, collaboration and other key skills. Coding gives students an opportunity to develop their creativity and express their identity by creating personally meaningful projects. At JCDS, young coders create programs to develop and express their Jewish identity in a pluralist community. The abilities to effectively use and create technology as an avenue for personal expression and to solve complex problems are the new and essential literacy skills of the twenty-first century. The vision of the JCDS Coding as Literacy Framework is to engage students in computer science skills and concepts through the integration of practices, while making connections to what they know and the world in which they live.

The purpose of the *JCDS Coding as Literacy Framework* is to provide a map as we develop and implement this innovative approach to coding as a language of expression in a Jewish context. These documents reflect an integration of national and state frameworks, grounded in our unique pluralist Jewish context. They were built using resources such as the *Massachusetts Digital Literacy and Computer Science (DLCS) Curriculum Framework,* the K-12 CS Frameworks, and the research of Prof. Marina Umaschi Bers at the Developmental Technologies Research Group at Tufts University.

The *JCDS Coding as Literacy Framework* was created using the following guiding principles:

- The 'playground approach' to coding, grounded on the **Positive Technological Development (PTD)** framework. PTD looks at the cognitive, personal, social, emotional, and moral dimensions of development through six positive behaviors facilitated by technologies: collaboration, communication, community building, creation, creativity and choices of conduct.
- Computer science education should focus on **powerful ideas of computational thinking:** algorithms, modularity, control structures, representation, hardware/software, design process, and debugging.
- **Coding is the new literacy.** Coding is more than a technical skill; it is a way to achieve literacy in the twenty-first century, like reading and writing. Computer science education should strive to give children the most powerful impact of computational literacy: expression with their own voices.
- All technology introduced to students must be grounded in **Developmentally Appropriate Practice.**
- Students learn best when solving authentic problems and engaged in **project-based learning**

For more information, please contact: Jared Matas, Director of STEM Innovation - jaredm@jcdsboston.org

	JCDS Coding as Literacy Frameworks: K - 2	
Powerful Ideas	**Concept**	**Skill:** *Student is able to ...*
Algorithms	People follow and create processes as part of daily life. These processes can be expressed as algorithms that computers can follow.	- Program syntactically correct complex sequences with 6 or more blocks without assistance. - Create an algorithm to solve a problem (e.g., move a character/robot/person through a maze).
Modularity	A computer program is a set of commands created by people to do something. Complex tasks can be broken down into simpler instructions, some of which can be broken down even further.	- Create a simple computer program. - Break up programming task into parts that are inter-dependent or recursive. - Use clusters of blocks as units in larger programs. - Combine pieces of different programs to achieve a goal.
Hardware / Software	Smart objects are not magical; objects are human engineered. A computing system is composed of hardware and software. Hardware consists of physical components, while software provides instructions for the system.	- Describe the function of electronics in a robot. - Understand that you must program a robot or computer in order for it to function. - Explain that computing devices function when applications, programs, or commands are executed.
Control Structure	Computers only follow the program's instructions. Computers follow precise sequences of instructions that automate tasks. Program execution can also be nonsequential by repeating patterns of instructions and using events to initiate instructions.	- Use control structures, including loops, event handlers, and conditionals to specify the flow of execution. - Correctly name and use parameter cards, loops and conditionals. - Purposefully use and understand the effect of a repeat loop. - Understand how control is passed through message blocks.
Debugging	Things do not just happen to work on the first try - many itierations are usually necessary to get it right.	- Recognize bugs in program sequences and is able to fix the problem using trial and error.
Design Process	The Engineering Design Process is an iterative process which provides steps for planning and creating human-made objects or processes to solve problems. The engineering design process includes identifying a problem, looking for ideas, developing solutions and sharing solutions with others.	- Self-initialize and effectively use the engineering design process.
Representation	Information in the real world can be represented in computer programs. Programs store and manipulate data, such as numbers, words, colors, and images. The type of data determines the actions and attributes associated with it.	- Identify all symbolic blocks, sensors, and parameter cards. - Explain use of blocks, sensors and parameter cards in relation to KIBO or more abstractly in relation to programming in general.

	JCDS Coding as Literacy Frameworks: 3 - 5	
Powerful Ideas	**Concept**	**Skill:** *Student is able to ...*
Algorithms	Different algorithms can achieve the same result. Some algorithms are more appropriate for a specific context than others.	- Generate multiple solutions for the same problem (or sub-problem). - Use logical reasoning to predict outcomes of an algorithm. - Write, debug, and correct programs using successively sophisticated techniques.
Modularity	Programs can be broken down into smaller parts to facilitate their design, implementation, and review. Programs can also be created by incorporating smaller portions of programs that have already been created.	- Combine pieces of different programs to achieve a goal. - 'Re-mix' code from others or their own programs in new ways.
Hardware / Software	Hardware and software work together as a system to accomplish tasks, such as sending, receiving, processing, and storing units of information as bits. Bits serve as the basic unit of data in computing systems and can represent a variety of information.	Describe the differences between hardware and software.
Control Structure	Control structures, including loops, event handlers, and conditionals, are used to specify the flow of execution. Conditionals selectively execute or skip instructions under different conditions.	- Use arithmetic operators, conditionals, and repetition in programs. - Understand and purposefully use all trigger and loop blocks.
Debugging	Debugging involves systematic analysis and evaluation using skills such as testing, logical thinking and problem solving in an intentional, iterative step-by-step way.	- Detect and correct logical errors in various algorithms. - Use interactive debugging to detect and correct simple program errors. - Start to be systematic in approach debugging.
Design Process	People develop programs using an iterative process involving design, implementation, and review. Design often involves reusing existing code or remixing other programs within a community. People continuously review whether programs work as expected, and they fix, or debug, parts that do not. Repeating these steps enables people to refine and improve programs.	- Design, build, and iterate a project independently.
Representation	Concepts can be represented by symbols, ie: letters represent sound, numbers represent quantities, programming instructions represent behaviors. Variables store values that represent data.	- Utilize variables successfully.

	JCDS Coding as Literacy Frameworks: 6 - 8	
Powerful Ideas	**Concept**	**Skill:** *Student is able to ...*
Algorithms	People design algorithms that are generalizable to many situations. Algorithms that are readable are easier to follow, test, and debug.	- Create a program that implements an algorithm to achieve a given goal. - Compare algorithms to solve a problem, based on a given criteria (e.g., time, resource, accessibility).
Modularity	Programs use procedures to organize code, hide implementation details, and make code easier to reuse. Procedures can be repurposed in new programs. Defining parameters for procedures can generalize behavior and increase reusability	- Decompose a problem and create a sub-solution for each of its parts (e.g., video game, robot obstacle course, making dinner). - Use functions to hide detail in a program.
Hardware / Software	Computing devices take many forms (e. g., car, insulin pump, or robot), not just personal computers, phones and tablets. They use many types of input data (collected via gesture, voice, movement, location, and other data) and run instructions in the form of programs to produce certain outputs (e.g., images, sounds, and actions). Hardware and software determine a computing system's capability to store and process information.	Identify and describe the use of sensors, actuators, and control systems in an embodied system (e.g., a robot, an e-textile, installation art, smart room).
Control Structure	Programmers select and combine control structures, such as loops, event handlers, and conditionals, to create more complex program behavior.	- Design and implement problem solutions using a programming language, including all of the following: looping behavior, conditional statements, expressions, variables, and functions.
Debugging	When code doesn't work as intended, trouble-shooting strategies applied in a systematic fashion can help.	- Trace programs step-by-step in order to predict their behavior. - Apply strategies to engage in systematic problem solving regardless of the programming platform. - Identify and fix errors using a systematic process.
Design Process	People design meaningful solutions for others by defining a problem's criteria and constraints, carefully considering the diverse needs and wants of the community, and testing whether criteria and constraints were met.	- Use an iterative approach to development and debugging to fully understand and address the dimensions of a problem.
Representation	Computational modeling and simulation help people to represent and understand complex processes and phenomena.	- Use, modify, and create computational models and simulations to analyze, identify patterns, and answer questions of real phenomena and hypothetical scenarios.