codeSpark ACADEMY

Introduction to Computer Science

Grades K-5

presented by codeSpark

HOUR OF CODE Edition

codeSpark

codeSpark Teacher's Guide

Dear Intrepid Teacher,

Thank you for your interest in teaching computer science to your kids! Knowledge of computer science and "algorithmic thinking" is increasingly necessary for success in our digital world. This skill is becoming a critical component of 21st century literacy. codeSpark created The Foos as an introduction to the "ABCs of Computer Science."

While it's important to prepare kids for the modern workplace, computer science is about much more than getting a job in high tech. Research shows that computer science helps students improve in core areas like math, logic, and even reading comprehension. Often people think of programming or coding as computer science but that is just one element. At its core, computer science is the study of how to use logical thinking to identify, simplify and solve complex problems. Not 0's and 1's.

Studies from MIT and Tufts University show that students as young as five can learn complex computer science concepts, especially when artificial barriers like programming syntax are out of the way.

codeSpark has created a unique and powerful approach to teaching computer science built on cutting edge research and hundreds of hours of prototype testing. codeSpark's learning games are designed with no words so even pre-readers and ELL students can play and learn from our powerful curriculum.

By playing our games, your students will improve their critical thinking skills, and improve in other disciplines, all while having a lot of fun!

-- The codeSpark Team

Have questions or feedback? Email us at info@codespark.org

Get our app here – http://thefoos.com

TABLE OF CONTENTS

codeSpark

4 Overview
10 Glossary
11 Game Kit Lesson
16 Unplugged Activity
18 Rubric

Download the full 10 lesson curriculum at thefoos.com/hourofcode

Teacher Overview:

Our lesson plans provide a fun, flexible and engaging introduction to foundational computer science concepts. They target students in K – 5th grade, but we've successfully tested with students as old as 8th grade.

All lesson plans are meant to be highly adaptable. You will be the best judge of what your students need to spend more time on and what they seem to enjoy the most.

In addition, all lessons include both an activity with our game, *codeSpark Academy with the Foos*, and an "unplugged" activity that does not require a computer or other connected device.

Materials:

- Our game, *codeSpark Academy with The Foos*. Download it at thefoos.com *Available for FREE on iPad, iPhone, Android Devices and web (e.g. major browsers Chrome, Safari, IE, etc.)*
- Props as needed: see lesson for specifics

No experience is necessary, but we recommend you review the lesson and play a few levels of the game prior to teaching it for the first time.

Note: This Hour of Code curriculum gives you approximately **2 one hour lessons.**

If you sign up for our "Foosletter" at thefoos.com, we will notify you when we expand the curriculum and release new versions of the game.

You can also download our **full 10 lesson curriculum** at thefoos.com/hourofcode.

OVERVIEW

What Is Computer Science?

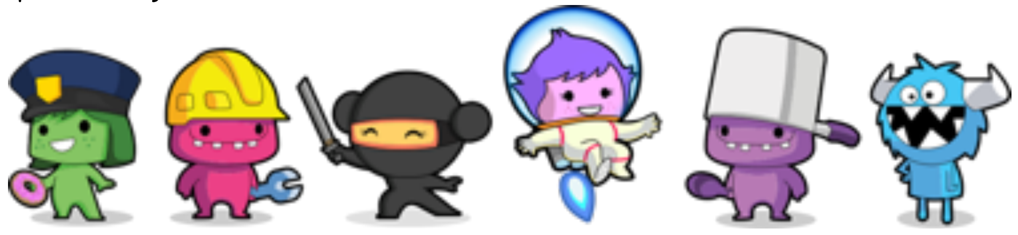
Computer Science, or “CS” boils down to solving problems with very specific sets of instructions because computers only do exactly what they are told to do. We think of computers as smart but in reality we need to tell them what to do! They can’t anticipate what we want from them; only computer scientists can come up with precise instructions computers need in order to act. Learning to think like a computer scientist or programmer helps children break down problems, think in logical sequences, and use precise language to give instructions.

The first lesson focuses on identifying common objects that only work when given the proper instructions. Then we will put this idea to work by programming the Police Foo – the first character players meet in our game.

OVERVIEW

Who Are The Foos?

The Foos are lovable and cute characters recently discovered by scientists. They are very small and live deep inside every computer, including smartphones, tablets and the computers in your class!



Each Foo can walk, jump, throw, eat and navigate their world, called "Fooville." Some Foos have special abilities that make them unique, for example:

- **Police Foo** - can chase and capture the Glitch
- **Chef Foo** - can make many kinds of food
- **Ninja Foo** - can shrink or grow bigger
- **Astronaut Foo** - can travel in four different directions
- **Construction Foo** - can make crates and also blow them up

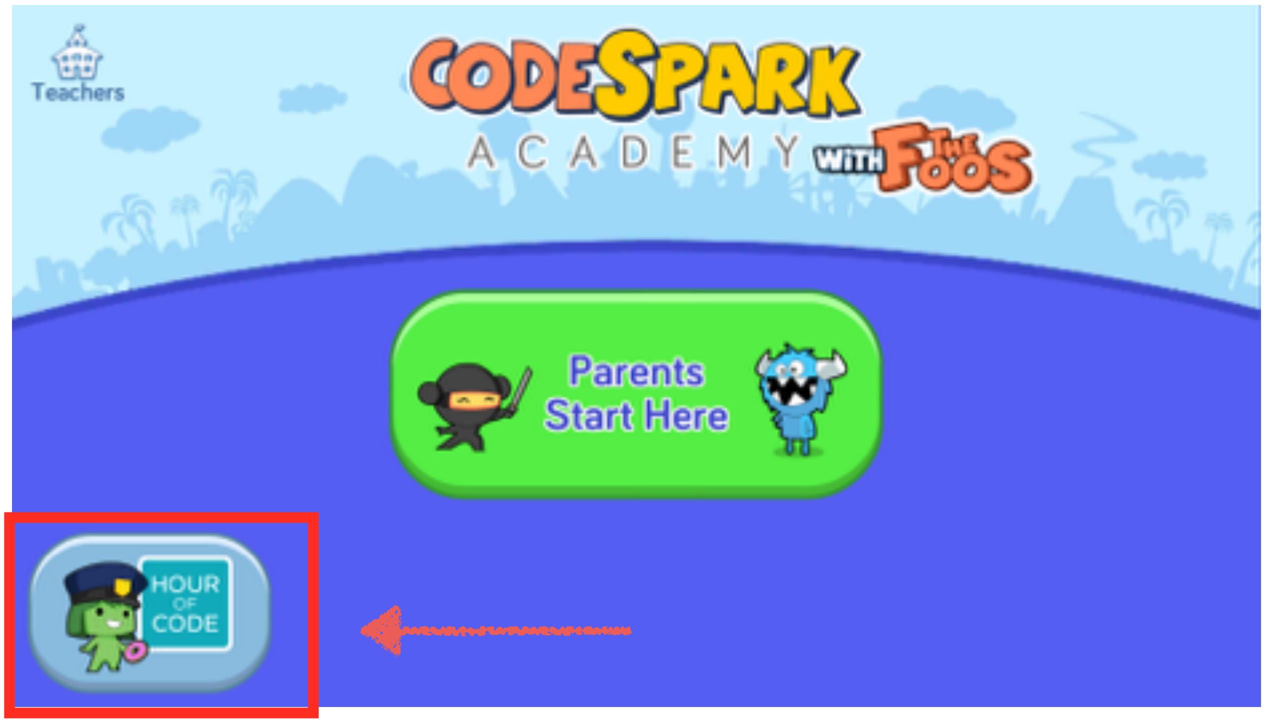
But, just like computers, the Foos only do what they are told. Students must learn to give them specific commands, or program the Foos, in a specific order.

Look out! The blue character with the white horns is the **Glitch**. He is a force of chaos in Fooville. Sometimes he makes a mess, sometimes he throws things around and sometimes he appears unexpectedly.

OVERVIEW

Tips and Tricks

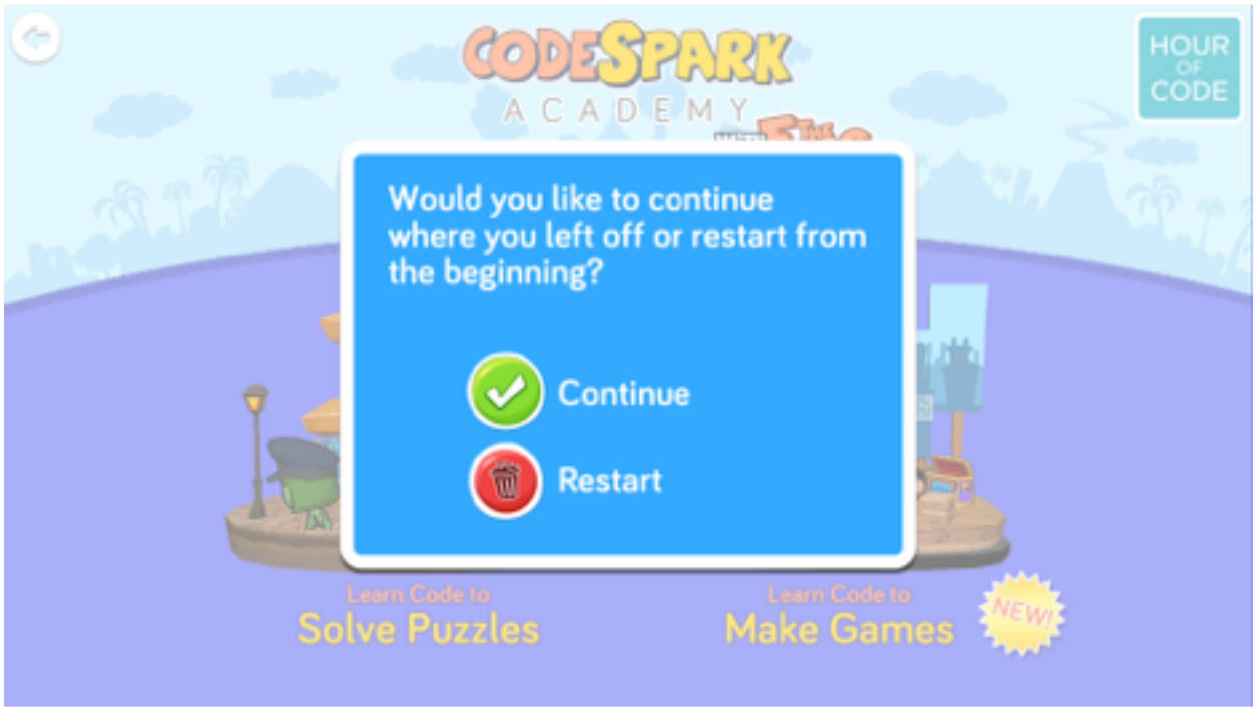
To launch the Hour of Code experience, press the “Hour of Code” button on the bottom left of the home screen.



On the next screen you can select which experience you’d like for your students: puzzles or the game creator.

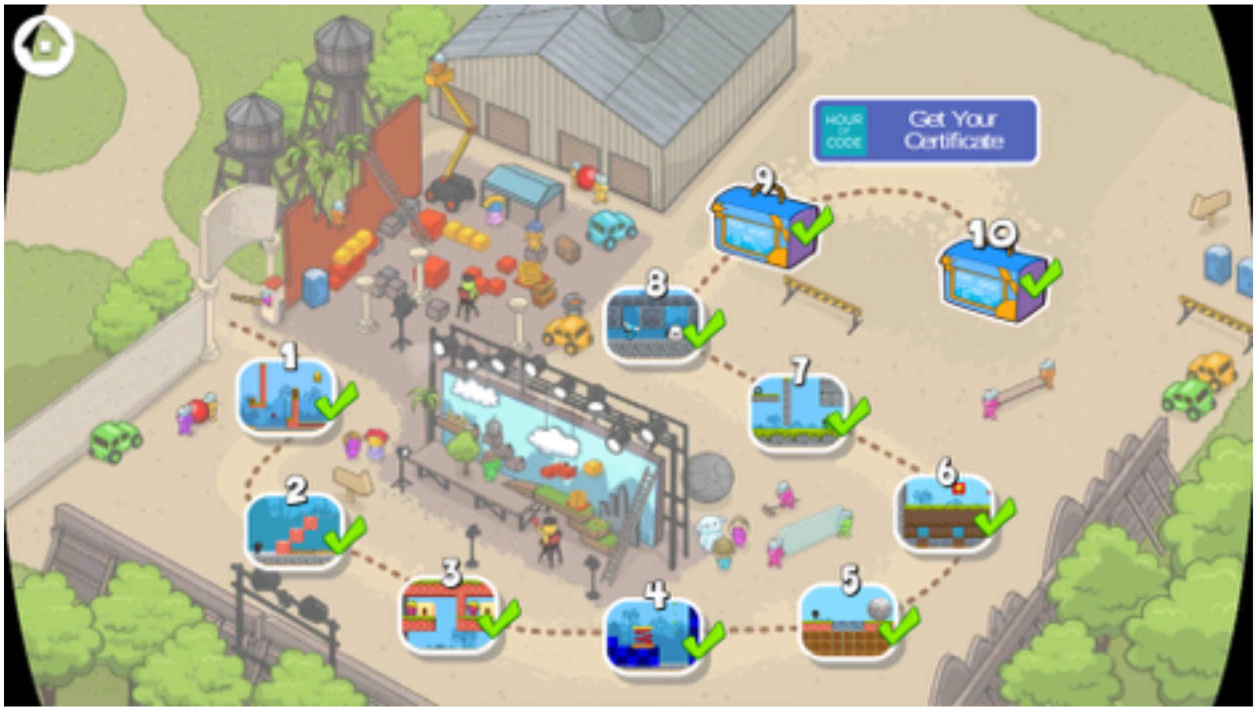
Reset your progress:

To reset a player's progress, exit and re-enter the Hour of Code experience. You should get a pop-up that asks if you'd like to clear your progress.


Would you like to continue where you left off or restart from the beginning?

OVERVIEW

What the Game Creator Experience?



The Game Creator is where players can build their own video game levels using 2 of our “game kits,” which are paint-by-numbers plans to creating your own game. Players can apply the coding principles they learned in the puzzle levels to reprogram any object in the Game Creator.

The first 8 levels of the Game Creator are tutorials to get students used to all of the Game Creator Tools. Level 9 and 10 are game kits.

GLOSSARY

codeSpark

Loop: A set of instructions that is repeated over and over again.

Endless Loop: A set of instructions that is repeated over and over again without end.

Remix: Building upon and adapting existing programming to create a new version.

GAME KIT LESSON
MAKE YOUR OWN GAME

codeSpark Academy with The Foos

codeSpark

Time:

45-60 Min

Materials:

Tablets or Computers with *codeSpark Academy*
Command and Parameters Dance Cards

Learning Goals:

Students will...

- Understand that some sequences are more efficient, and thus more desirable, than others.
- Loops make sequences more efficient.
- Efficiency is important because computers don't have unlimited processing power.
- Understand the difference between a loop and an endless or infinite loop

Vocabulary:

Loop: A set of instructions that is repeated over and over again.

Endless Loop: A set of instructions that is repeated over and over again without end.

Remix: Building upon and adapting existing programming to create a new version.

GAME KIT LESSON
MAKE YOUR OWN GAME

codeSpark Academy with The Foos

codeSpark

Introduction:

If you have a projector or Smartboard, open up codeSpark Academy and the Construction Foo level 10 for the whole class. This is where loops are first introduced. Point out the loop symbol in the bottom right hand corner and play through the level to show students how loops work. Ask students to help you figure out how many times you want Construction Foo to repeat an action. Continue playing through the Construction Foo levels and show students what happens if the loop command is set to too few or too many repetitions.

Ask students, what if you wanted an action to go on and on without ending?

Introduce the idea of endless loops and provide some examples, such as the earth rotating around the sun, time, and electricity. Draw the infinity symbol ∞ on the board, which is used in the Foos to represent endless loops.

Game Activities:

Have students play through the first nine levels of the "Make Games" section. These tutorial levels will help students become familiar with the different game design components and how to use and adapt them to create their own games (overview provided below).

Overview of tutorial levels 1-8

Level 1: Learn how to play through a game
Level 2: Learn how to add scenery (e.g., bricks)
Level 3: Learn how to make sprites "walk forward"
Level 4: Learn how to make sprites jump
Level 5: Learn how to erase game components
Level 6: Learn how to explode objects and scenery
Level 7: Learn how to make sprites and objects grow
Level 8: Learn how to use loops to make actions repeat

GAME KiT LESSON
MAKE YOUR OWN GAME

codeSpark Academy with The Foos

codeSpark

Once students play through the nine tutorial levels, two treasure chests (Levels 9 and 10) will unlock. Both levels begin with a short video showing an overview of what the game will look like once it is made, followed by interactive instructions on how to program a particular game mechanic using different computer science concepts.

Have students work individually or in partners to work through both levels. Both levels provide "game kits" that outline the different game components they need to add to make the game. If students work in pairs, make sure they take turns both creating the game components and playing the game. This could include students working together on both components or one student takes on the "programmer" role to create the game and the other plays it and then they switch roles.

After students finish making their games according to the blueprints, challenge them to adjust different game components and make it their own. Below are some example challenges.

Example Challenges

- What happens if you change the infinity symbol to a number when using loops?
- How can you make your game more challenging? Try adding different game mechanics to see what happens!
- If working in programmer/player pairs, encourage students to remix their partners' game by changing certain elements (e.g., where the enemy sprites are placed, the direction the sprites jump, adding additional elements that the Foo must avoid).

Unplugged Activities:

Dancing Loops

1. Divide students into pairs and provide each pair with Command and Parameters Dance Cards.

2. In pairs, have students use the cards to create their own dances. There are two rules for dance:
    - The dance must be contained within an endless loop.
    - It must include at least one command and one parameter.

3. Have pairs write down the sequence of actions and the parameters that make up their dance.

4. Have each pair present their dance to the whole class (act it out!) and ask the other students to identify what were the commands and parameters of the dance loop.

Command and Parameters Dance Cards available in the back of the book.

Debrief Discussion:

- What are the benefits of using loops? (hint: they're more efficient)
- What are some real-life situations where endless loops would be more beneficial than regular loops? What are situations where regular loops might be better?
- Examples: treadmills, escalators, turn signals, water cycle

ProTip: *Debugging* - Often in computer science, we encounter mistakes that make our programs do things incorrectly. When creating their dance, students might have made mistakes in their code. Remind students that making mistakes is part of the process, and we can learn from every mistake.

UNPLUGGED ACTIVITY

codeSpark

Endless Loop Dance Activity

Commands & Parameters Cards

Grey cards are **Commands** while white cards are **Parameters**

CLAP	TURN AROUND	ONCE	TWICE
SNAP FINGERS	WAVE HAND	THREE TIMES	RIGHT
JUMP	SHAKE HIPS	LEFT	UP
HOP	WIGGLE	DOWN	FAST
WADDLE	TAP FOOT	SLOW	FOUR TIMES

RUBRIC FOR STUDENT EVALUATION

	Unsatisfactory	Competent	Proficient	Excellent
Concepts	Puzzle levels are not completed.	Puzzle levels are completed with 1 star.	Puzzle levels completed with 2 stars.	Puzzle levels completed with 3 stars.
Execution	Code does not work or has major flaws preventing it from working correctly.	Code mostly works, or has minor flaws.	Code works in the way the student intended but is not the most efficient.	Program is functional, refined, and is executed in the most efficient way possible.
Grasp of Materials	Student cannot describe how their code should work and are unaware of their process.	Student can mostly describe how their code should work and some understanding of content.	Student can describe how their code should work and troubleshoot problems preventing their desired results.	Student can describe how their code works, how they wrote it, and help others troubleshoot their code.
Effort	Student shows minimal effort, does not use class time effectively, and work is incomplete. Student refuses to explore more than one idea.	Student does enough to meet minimum requirements. Student has more than one idea but does not pursue.	Completed work in an above average manner, although more could have been done. Student explores multiple solutions.	Completed work and exceeded teacher expectations. Student displays willingness to explore multiple ideas and solutions and asks questions.

Rubric adapted from: http://www.edutopia.org/pdfs/blogs/edutopia-yokana-maker-rubric.pdf

DOWNLOAD THE FULL CURRICULUM AT THEFOOS.COM/HOUROFCODE