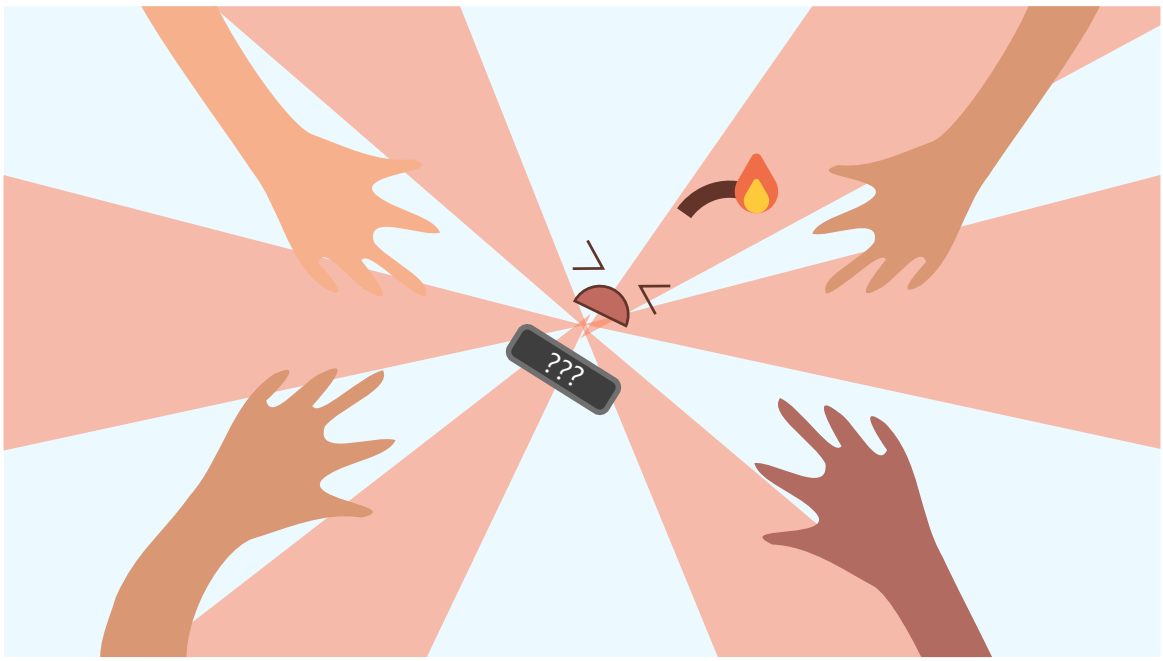

The Digital Hot Potato

Code a suspenseful game with an unpredictable timer. Learn how to use the 'while' loop for actions that must repeat an unknown number of times.

Courses

- Grades 6-12

Materials

- Mobile phone, tablet, or computer
- Internet connection
- Cardboard, scissors and a couple of rubber bands to hold the phone against the cardboard

Description

This activity introduces the "while" loop, the fundamental loop structure for situations where the number of repetitions is unknown. Students will build a "hot potato" game with a random timer, a perfect scenario that shows why we need a loop that runs *while* a condition is true, instead of a fixed number of times.

Educational Objectives

- Understand the structure and purpose of a "while" loop.
- Clearly tell apart when to use a "for" loop and when to use a "while" loop.
- Use random numbers to create unpredictable behavior in a program.
- Manage the flow of a program that depends on a condition that changes while it is running.

Start (10 minutes) - The Mystery Loop

1. Welcome the class: **"We've learned the 'for' loop, which is like running a 100-meter race: you know exactly where the finish line is. But what if you're in a maze? You don't know how many steps it will take to get out; you only know that you have to keep walking *while* you haven't found the exit."**

2. Introduce the concept: **"That 'while' is the key to our last type of loop: the 'while' loop. It doesn't run a set number of times, it runs as long as a condition is true."**
3. Present the game: **"Today we'll use this idea to create a game of 'hot potato'. The timer will be random, so we don't know when it will end. The game keeps going *while* the remaining time is greater than zero."**

When You Don't Know When to Stop

The `for` loop is great when you know the exact number of repetitions in advance (for example, "repeat 10 times" or "go through the 7 pixels in a row"). But in many cases, like a game with a random timer or waiting for someone to press a key, you have no idea how long it will take. You need a loop that can keep running **while** a condition is true, no matter how many repetitions that turns out to be.

The 'While' Loop (Repeat While)

The **"while" loop** (or "repeat while") is the perfect solution for these cases. It works with simple logic: 1. First, it checks a condition (for example, `is timeLeft > 0?`). 2. **While** that condition is **true**, it runs the code inside it over and over again. 3. On every repetition, it goes back to step 1 and checks the condition again. 4. The moment the condition becomes **false**, the loop stops immediately and the program moves on to the next block.

The Logic of the Hot Potato

Our game combines several ideas you already know in a brand new way: - **Event:** Pressing the screen starts a new round — if there isn't one already running. - **Flag:** The `ready` variable remembers whether the game is free. It starts out true in `when the program starts`; the event only starts a round `if ready`, and the first thing it does is set it to false, so a second touch can't restart the round in progress. - **Variable and Randomness:** The `timeLeft` variable stores a **random** number so that every round is unpredictable. - **While Loop:** The main game loop runs **while** **`timeLeft > 0`**. Inside the loop,

we subtract 1 from `timeLeft` every second. - **The End:** When `timeLeft` reaches 0, the loop's condition becomes false, the loop finishes, and the potato "explodes"!

Development (20-30 minutes) - Coding the Suspense

1. Now that students understand the conceptual difference between `for` and `while`, it's time to build the game.
2. Guide them through **the instructions for creating the game, focusing on the structure of the 'while' loop and its stopping condition (`timeLeft > 0`)**, as detailed below. It's crucial that they understand why this loop is the right tool for this job.

Closing (5-10 minutes) - The Right Tool for the Job

1. Once everyone is passing the "hot potato" around, it's time for the final reflection on loops.
2. Start the discussion: **"Let's compare the tools we know. When would you use a 'for' loop? And when would you use a 'while' loop? Why was a 'while' loop perfect for this game, and why wouldn't a 'for' loop have worked as well?"** (Because the remaining time is random and unknown when the loop starts).

Reflect

What is the main difference between a "for" loop and a "while" loop? Why was "while" the best choice for this game?

Inside a "while" loop, it's really important to make sure the condition eventually becomes false, so you don't get stuck in an infinite loop. Which block inside our loop guarantees that the game always ends?

What would happen if the random number could come out as 0? Would the game still work properly?