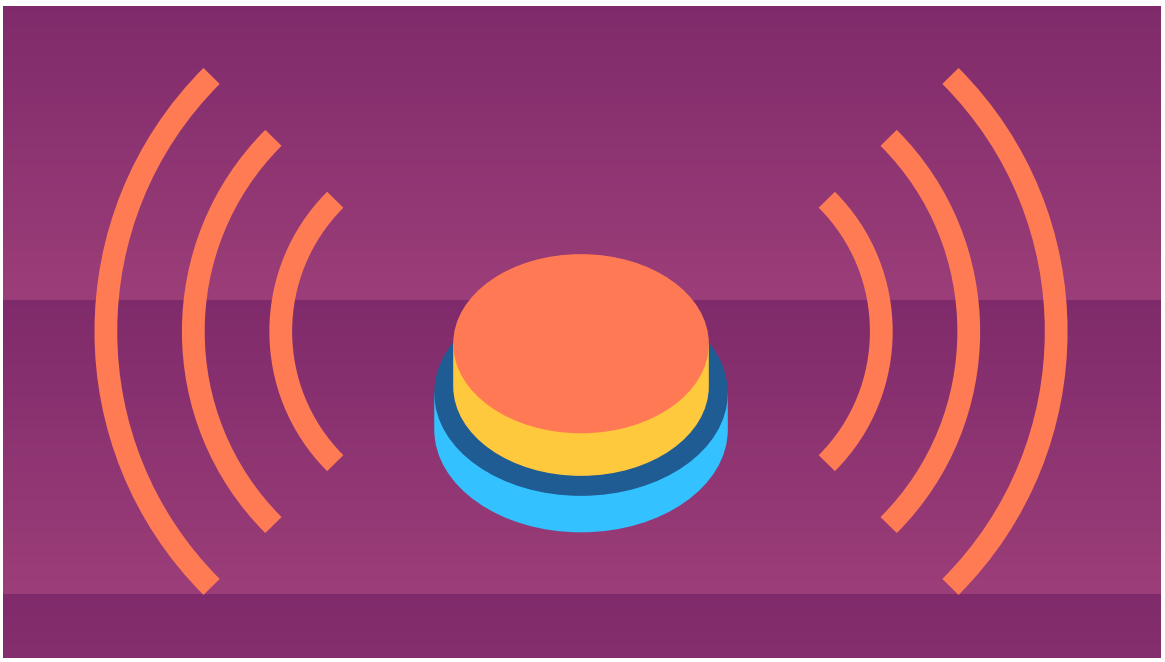

A Smarter Alarm with Events

Discover a more efficient way to code! Learn how events let your program react instantly to actions like a button press.

Courses

- Grades 3-12

Materials

- Cellphone, tablet, or computer
- Internet connection

Educational Objectives

- Understand the concept of an event.
- Create a technological object (prototype) using a device.
- Identify relationships between technology and the surrounding world.
- Evaluate personal and others' work.
- Engage in dialogue and reflection on improvement ideas.

Start (10 minutes) - A Smarter Way to Listen

1. Welcome students and introduce the day's activity: **"Today, we will learn how to create a simple alarm that plays a sound when you press a button, but using a new, smarter method."**
2. Start by asking the class: **"How does a simple button or switch actually work?"** Guide them to the idea of two states: ON and OFF.
3. Then, introduce the core concept for today: efficiency. Ask them, **"Using a loop, our program has to constantly ask the button, 'Are you pressed? Are you pressed?'. What if the button could just send a message to the program *only* when it's actually pressed?"** This introduces the concept of an **event** as a notification—a more efficient way to code.

How does a switch work?

We are going to create an alarm that is activated by a switch. But to do that, we need to understand how a switch thinks. In programming, a simple switch has two possible **states**: * **On:** The switch is activated. We can represent this

state with the number **1**. * **Off:** The switch is deactivated. We can represent this with the number **0**. A switch can only be in one state at a time. By reading whether the state is 1 or 0, the computer can decide what to do next.

How does the computer know the state?

The computer can be programmed to "listen" for a signal from the switch. **Every time the button's state changes (from Off to On, or On to Off), it notifies the computer!** This notification is called an **event**. Once the computer receives the event, it can run a specific piece of code to react to it.

Why not just use a main loop?

Using a **Main Loop** to check a button is like a backseat driver constantly asking: *"Are you pressed yet?! How about now?! Are you pressed now?!"* even when nothing is happening. It's repetitive and wastes the computer's energy. With **events**, the program is quiet. It just waits. The switch itself sends a single signal to the computer *only when its state has changed*. This is much more efficient and professional!

We received the event... Now what?

Once the switch sends its signal, the computer can use a conditional to decide what to do based on the switch's current state. For our alarm, the logic is simple: **IF** the switch's new state is **ON**, we play a sound. **ELSE** (meaning its new state is **OFF**), we stop the sound.

Development (20-30 minutes) - Building an Event-Driven Alarm

1. Now that the students understand the difference between constantly checking with a loop and efficiently listening for an event, it's time to build the smarter alarm.
2. Lead them through **the instructions for building the event-driven program**, as detailed in the hands-on section below. Emphasize that this program's main logic does **not** use a "repeat forever" loop, because the event does the work for us.

Closure (5-10 minutes) - Events are Everywhere

1. Once everyone has a working, event-driven alarm, it's time to reflect on this powerful new programming paradigm.
2. Use the final section to spark a discussion on where else events are used in modern technology (every click, tap, and notification is an event!) and to challenge them to combine events with variables to add a new feature.

Reflect

You've just learned a more professional way to code!

Events are the backbone of all modern user interfaces—every click, tap, and swipe is an event. *Besides buttons, what other things on your phone or in a game could trigger an event? (Think about notifications, finishing a download, a character touching a coin, etc.)*