Lesson 29

technology

Scratch: loops, events and finding the bug

Lesson at a glance

Open by framing the class as programmers building Frog Frenzy: a frog that pops up at random for pupils to click and score. Model each Scratch piece on the IWB—hide, a forever loop with wait/go-to/show/hide, a score variable, and a when-this-sprite-clicked event—then pairs build and run after every step. Debug common slips (blocks outside the loop, code on the wrong sprite) as they appear. Close with a display-only code talk on bugs found and one change they would make.

Learning objectives

- Add a loop and an event to a Scratch program so a sprite responds as planned
- Find a bug when the program goes wrong and fix it
- Explain in their own words what the loop or event does

Before the bell — prep

Book devices with Scratch 3 ready and sound enabled. Open a fresh project on the IWB and confirm Forest backdrop and Frog sprite are in the library. Pair pupils at machines before the bell so login time does not eat the build.

Materials

Item	Qty	Per	Source	Low-cost substitute
tablet or laptop with Scratch	1	pair	school devices	run the lesson on the IWB with pupils taking turns

Safety watch-point

Manage volume on shared devices so the pop sound stays comfortable; remind pairs to carry laptops with lids closed between desks.

Lesson 29

Teaching moves

- **What we're building:** Set the watch-then-do routine on the IWB: you model, they build, they run the green flag. Ask what score-keeping games they know, then name bugs as normal coder work so pairs keep testing without freezing.
- **Create a New Project:** Model new project and delete the cat. Circulate until every pair has a blank stage—stop anyone still editing the cat before the next step.
- **Add the Forest Backdrop:** Show the backdrop button versus the sprite button on the board. If a pair adds Forest as a sprite, have them delete it and choose backdrop instead.
- **Add the Frog Sprite:** Model adding Frog, then ask 'Which sprite is highlighted?' before any code. Blocks on the wrong sprite is the early bug that blocks the whole game.
- **Make the Frog Disappear:** Snap when-green-flag-clicked and hide on the IWB, name it an event, and have pairs run the flag. Ask what the flag just did; watch for pairs who never click it and think the code failed.
- **Random Frog Appearance:** Zoom so the class sees wait, go-to random, and show sitting inside the forever mouth—not stacked below it. Ask why the frog keeps reappearing, then support pairs block-by-block as they nest the loop.
- **Hide the Frog again:** Have pairs predict the new rhythm before running. Add wait 2 seconds and hide inside the loop only; stop the class if those blocks land outside forever and re-nest them together.
- **Create a Score Variable:** Model Make a Variable named score first, then set score to 0. Ask what the scoreboard will count and point out the stage readout so pairs know it is live.
- **Score a Point:** Model when-this-sprite-clicked with change score by 1, pop sound, and pixelate. Contrast it with the green-flag event. Ensure sound is on and that pupils click the frog, not the stage.
- **Clear the Effect:** Place clear graphic effects at the top inside forever, before wait/show. Ask why we clear before the next appearance, then let pairs play and chase a high score.
- **Code talk — what worked and what we'd change:** Display-only talk—no typing. Invite a pair who hit a bug to describe how they spotted and fixed it; revoice that running the flag often is how coders find problems. Ask one change they would make next time.

What it should show

On green flag, score resets to 0; the frog hides, waits 1 s, jumps to a random spot, shows for 2 s, hides, and repeats. A click on the visible frog adds 1 to score, plays pop, and pixelates until the next loop clears effects. If the frog never moves, blocks are usually outside forever or on the wrong sprite; if it stays pixelated, clear graphic effects is missing or misplaced.

Misconceptions & interventions

- **Pupils stack wait, go-to, show and hide under the forever block instead of inside its mouth, so the loop runs empty and the frog never pops.** — Zoom the IWB on the C-shaped forever and drag one block in so they see the nest. Have them run the flag and describe the appear–vanish rhythm.
- **Pupils add code to the stage or a leftover cat because the frog is not the selected sprite.** — Ask 'Which sprite is highlighted?' and point at the sprite pane. Delete stray scripts and rebuild on the frog only.
- **Pupils think the score or pop should happen on green flag, or they click the stage and expect points.** — Hold up when-this-sprite-clicked versus when-green-flag-clicked and trace: scoring runs only on a frog click. Have them click the frog body once it shows.

Differentiation

Emerging	Developing	Proficient
• Sit at the teacher table and build one script at a time from the IWB model; teacher checks the frog is selected and the forever nest before they run the flag. • Offer a printed block order (hide → forever → wait → go to → show → wait → hide) they can tick as each piece is placed.	• After the game works, change wait times (e.g. shorter show) and re-run to feel the difficulty shift. • Prompt: 'In your own words, what does the forever block make the frog do?'	• Critique fairness of the click window and suggest one rule change (wait times or an extra sprite) without leaving Scratch. • Explain to another pair how the click event differs from the green-flag event and where they would hunt a scoring bug first.

Cross-curricular hook

Link to Maths Data: pairs note scores across short plays and compare who scored most and the score range.