LESSON 5

Testing and debugging in Scratch

40 minutes | Strand 1: Introducing computer science | Built in Scratch | Outcome 1.8

WHAT THIS LESSON DOES

Students build a Racing Car project in Scratch and practise testing it to check it works, then find and fix bugs using a named routine: read the symptom, find the block, form a hypothesis, and test the fix.

CURRICULUM OUTCOMES

Outcome	What the specification asks for
1.8	test the code to determine its functionality and to identify and manage errors

WHAT STUDENTS SHOULD BE ABLE TO DO

- Test a program to check it works (LO 1.8)
- Find and fix a bug using a named routine: read the symptom, find the block, form a hypothesis, test it (LO 1.8)

BEFORE THE BELL

- Have devices ready with browsers open and students logged in to Scratch (scratch.mit.edu).
- Check that students can create a new project and that image download or upload is allowed on the school network for the car sprite file.
- Open a blank Scratch project on the board so you can model the first steps before pairs work independently.

WHAT YOU NEED

Item	Qty	Per	If you do not have it
tablet or laptop with Scratch	1	pair	run the lesson on the IWB with students taking turns

Testing and debugging in Scratch

HOW THE LESSON RUNS

3 min	introduction	Start: what we're building
2 min	content	Create a new Scratch project
2 min	content	Add the car sprite
2 min	content	Shrink the car
3 min	content	Draw a racing track
3 min	content	Place your car on the track
2 min	content	Create a speed variable
3 min	content	Detect where the car is
3 min	content	Move forwards and backwards
3 min	content	Move left and right
2 min	content	Drive your car!
6 min	content	Break it, then fix it with the routine
3 min	content	Make sense: what worked, what broke
2 min	content	Reflect
1 min	summary	What you covered

TEACHING MOVES

- Model the named debugging routine out loud when a bug appears: name the symptom, point to the likely block, state a hypothesis, then test the change.
- Remind students to place the car on the track first so the go to x y values prefill with a useful start position.
- When they set the touching-colour condition, have them use the colour picker on their own grass green so the sense matches their backdrop.
- Ask for a short prediction if you have time for one, then have pupils run the project after each control or sensing change rather than only at the end, so a bug is never more than one change old when they meet it.
- If you are running short: accept a rough track and trim the track and colour work at steps 5 and 6, or draw one on the board for pairs to copy. Leave the time for Break it, then fix it with the routine untouched, because that is where pupils actually test a program and fix a bug with the four steps, and a lesson that builds a car and never breaks it leaves that out.

WHAT TO WATCH FOR

- Touching-colour green does not match the grass, so speed never drops: reopen the colour picker and sample the backdrop green again.
- Go to x y was added before the car was placed, so it jumps to the wrong start: move the car on the stage, then replace or edit those values from the prefilled position.
- Arrow-key scripts sit outside the forever loop or use the wrong key, so one direction does nothing: check each key block is inside the forever and matches up, down, left, or right.

DIFFERENTIATION

- Support: Draw a simple track on the board for pairs to copy, or let them race on a plain backdrop with no grass at all, then sit with them to walk the four debugging steps on one broken arrow key. Decide which of those you are using before the bell.
- Extension: Challenge early finishers to tune turn angle or off-track speed and retest until the drive feels fair.
- Extension: Ask confident pairs to plant one deliberate bug for another pair and coach them through the named routine to find it.