

Week 2 — Teacher Run Sheet

Friday, August 21, 2026 · 45 minutes · Print and keep in your hand. The timing table is on page 1 — that's the only page you need mid-class.

Timing here is the *adjusted* pacing

Week 1 established that this class is already fluent in block coding. The mini-lesson below is compressed to 5 minutes and the build is extended to 27, which is **not** the split printed in the lesson page. Teach this version.

Before students arrive

- ✓ Projector on, `arcade.makecode.com` open, signed out
- ✓ **New Project** already created and named `Week 2 Move` — so your first clicks are visible, not fumbling
- ✓ Cross-age pairs posted, built from the Week 1 surveys (🕶️ with 🦖, chosen for **patience, not speed**)
- ✓ One blank sheet + index cards ready for Plan B
- ✓ Board: **Last Friday: Computers follow instructions exactly / Today: Sequence = now · Event = later / Pair rule: Driver touches the keyboard. Navigator catches the plan.**

No student logins today. MakeCode autosaves in the browser.

The 45 minutes

Clock	What	Watch for
:00	Anchor. Screens half-closed. "Some instructions run now. Some wait until something happens." Quick vote: <code>set background, make player, move player</code> — <i>which runs first? Point to it.</i>	Retrieval after a 7-day gap. Keep it under 4 min
:04	Mini-lesson — build 4 blocks fast. <code>set background → set mySprite to Player → controller move with buttons → on A pressed → say "Hi!"</code>	 Compressed on purpose. Don't narrate the block hunt — they know where blocks live

Clock	What	Watch for
:07	The one idea. "These three run once, top to bottom. This A block is different — the computer says <i>'got it, if A ever happens I know what to do.'</i> Might run ten times. Might run zero." Then Blocks → JavaScript for 20 seconds	This is the whole lesson. Everything else is practice
:09	Paired build — Moving Hero . New project. Background, player sprite, controller movement, one button event	Circulate. Ask "which block waits?" — never fix it
:22	 SWAP. "Freeze. Driver's hands off. Navigator becomes Driver."	⚠️ Enforce this out loud. It will not happen on its own
:36	Share-out. One simple working project, one interesting broken one. "Show us the exact event. What has to happen before that code runs?"	If it's broken: <i>"Sequence problem or event problem? What's your evidence?"</i>
:42	Teaser + Think About This	See below
:45	Done	

Tiers — the floor moved up this year

	Who	Build
 Starter	<i>Scaffold only</i> — whoever needs it	Sprite, background, speed, A-button message
 Builder	Everyone. This is the required build	Two button events. A speeds up, B slows down — or each changes the sprite differently
 Extender	Expected, grades 8+	Blocks → JavaScript. Find <code>controller.A.onEvent</code> . Change speed or text <i>in code</i> , toggle back, confirm it still runs
 Stretch	Anyone who lands  early	Delete <code>controller move mySprite with buttons</code> . Rebuild movement by hand from four button events — up, down, left, right — each setting <code>vx / vy</code>

✓ Why the stretch is the good one

It reveals that the convenience block was hiding four event handlers the whole time. That lands the day's concept harder than any explanation you could give — and it costs you nothing to assign.

Circulation questions — ask, don't fix

- "Which block runs first when the game starts?"
- "Which block is *waiting*?"
- "If you never press A, does that code run?"
- "Change one number. What changed on screen?"

For a pair that's already done at :20: *"Does that code still exist if you never press A?"* Fluency with blocks is not understanding of events — press there.

Think About This (read aloud at :42)

This week, catch a machine **waiting**. A microwave button. A crosswalk button. A game controller. A car sensor. A phone notification.

Ask yourself: **what was it doing before the event happened?**

Nothing to write down. Bring one example next Friday.

Teaser: *"Next Friday we teach the computer to do something again and again — without copying the same block ten times. That's where games start feeling alive."*

If the internet is down

Same concept, no devices. Index cards only.

Clock	What
:00	Same anchor
:04	Human program. Three cards on the board: <i>Stand up</i> · <i>Turn around</i> · <i>Sit down</i> . Student robot runs them in order. Swap two cards, rerun. Sequence is now visible
:12	Event cards. Add <i>WHEN</i> cards: <i>When teacher claps, wave</i> · <i>When someone says GO, jump once</i> . The robot ignores these until the event fires. That contrast is the lesson
:22	Pairs write a six-card classroom game — three setup cards, three event cards, each one labelled
:34	Trade programs. One pair robots, the other triggers events. Debug unclear instructions out loud
:41	Same teaser and Think About This

Rule stays strict: robots do what the cards say, not what the writer meant.

Troubleshooting

If...	Then...
Sprite appears but won't move	<code>controller move</code> is pointing at a different variable than the sprite they made
A-button event fires only once	Correct. <code>on A pressed</code> runs once per press, not while held
Pair finishes the required build in 8 min	Expected with this class. Send them to ● — do <i>not</i> hand out a bigger game
Older student has taken the keyboard	Enforce the :22 swap. Give the younger one a concrete Driver task: change the speed
"JavaScript is a different project"	Toggle back and forth on their screen . "Same game, two views"
Student opens last week's Chase the Pizza	Fine — but start a new project, or today's concept stays buried in the tutorial
MakeCode prompts for sign-in	Ignore it. Nothing today needs an account

After class — 5 minutes, same evening

- Which pairs got setup vs event **without** prompting? → those are your Week 3 loop leaders
- Who should be a ● Extender by default from now on?
- Did the :22 swap actually happen, or did you let it slide?
- What MakeCode friction is worth pre-demoing next Friday?