

Control flow that matches the design

Meeting 3 | Mappings, practice feedback, bounded loops, and conditional test

Bartosz Maćkiewicz

Programming Behavioral Experiments in jsPsych

Build a task that takes the right route after success, failure, or a change of response mapping.

Artifact: a flanker pilot with bounded practice and a test gate.

Start from the participant's journey

Instructions → practice → check the **current attempt**.

A bounded loop and a test gate solve different problems.

What is the correct answer?

←←←←←	left
→→→→→	right
←←→←←	right
→→←→→	left

Respond to the **central** arrow. Ignore the neighbors. Check every answer key independently.

A mapping is not a direction

Group A: left = F; right = J. Group B: left = J; right = F.

Store the target direction and the expected key separately. Instructions, scoring, and saved group must agree.

Choose a response mapping

```
const requested = new URLSearchParams(location.search)
  .get("group");
const group = requested === "B" ? "B" : "A";
```

Use `?group=B` for the reversed mapping; otherwise use A. The example reads one parameter, then uses one mapping.

Counterbalancing needs an allocation rule

A URL parameter carries an assignment; it does not create balanced groups by itself.

For a classroom demonstration, allocate A/B deliberately. A real study needs a documented allocation procedure and checks of actual group counts.

Label phases explicitly

```
const practiceItems = PRACTICE_ITEMS.map((item) =>
  ({ ...item, block: "practice" }));
```

Record task, block, group, condition, item, and attempt. Avoid inferring a phase later from row position alone.

A practice procedure

```
const practiceBlock = {  
  timeline: [fixation, trial, feedback],  
  timeline_variables: practiceItems,  
  randomize_order: true  
};
```

The test procedure uses [fixation, trial].

Feedback reads the last task response

```
const last = jsPsych.data.get()  
  .filter({ task: "flanker" })  
  .last(1).values()[0];
```

Distinguish correct, incorrect, and no response. Do not accidentally read a fixation or feedback row.

Live checkpoint L1: feedback only in practice

Add feedback to the practice procedure. Test a correct answer, a wrong answer, and a timeout.

Check that feedback has its own task label and does not enter the accuracy denominator.

Measure the current attempt

```
const recent = jsPsych.data.get().filter({  
  task: "flanker", block: "practice",  
  practice_attempt: practiceAttempts  
});
```

Pooling all previous attempts can keep a successful learner below criterion unnecessarily.

A bounded loop

```
const loopedPractice = {  
  timeline: [practiceBlock, practiceSummary],  
  loop_function: () => {  
    const repeat = !practicePassed && practiceAttempts < 3;  
    if (repeat) practiceAttempts++;  
    return repeat;  
  }  
};
```

Both the criterion and the maximum matter.

The threshold is a boundary case

Four trials, criterion 75%.

- 2 correct: fail.
- 3 correct: pass.
- 4 correct: pass.

Use ≥ 0.75 , not > 0.75 . Test the boundary explicitly.

Live checkpoints L2 and L3

L2: run both `?group=A` and `?group=B`.

L3: implement bounded repetition and inspect the attempt ID.

Use the arrow on screen to decide the correct key; do not let the saved answer key be your test oracle.

Conditional entry to test

```
const testIfPassed = {  
  timeline: [beginTest, testBlock, summary],  
  conditional_function: () => practicePassed  
};
```

Stopping the loop after three failures is not permission to enter test.

Exiting a loop after the final allowed attempt does not imply a pass. The conditional gate must still check practicePassed.

The failed-practice route

Show an explanatory stop screen and allow data export.

Tell the participant what happened and how to get help. No test rows should be created on this route.

The task must not loop forever or silently lower the criterion.

First platform step: reuse a working task

Keep the Meeting 2 experiment unchanged while learning JATOS.

- Start the prepared local JATOS installation.
- Import the supplied study archive.
- Run the study and retrieve its results.

Follow the local JATOS worksheet; setup trouble has a demo fallback.

Move a study, then run it again

Export a **.jzip**; import it into a different instance; run and retrieve a new pilot.

Study archives do **not** transfer previous results.

Three files, three purposes

File	Purpose
Source ZIP	Edit HTML, JavaScript, styles, and assets
Study .jzip	Import a configured study into JATOS
Results export	Inspect the recorded runs

Cognition.run is an alternative with its own editor and saving.

1. Check the mapping and item keys.
2. Add practice-only feedback and bounded repetition.
3. Gate test on passing practice.
4. Verify normal and failure routes.

The starter contains the structure; complete the marked parts.

Expected task-row counts

Route	Practice	Test
Pass first attempt	4	8
Fail, then pass	8	8
Fail all three	12	0

Fixation, instructions, and feedback are additional rows.

Excluding people who fail practice changes who reaches test.

Document the criterion, maximum attempts, and number stopped. Do not use a practice rule as an invisible post-hoc exclusion.

Complete the paper-to-prototype worksheet.

Name one manipulation, one response, one data row, and one failure route. Choose the smallest coherent version.

Proposal and Assignment 3 are due before Meeting 4.

Explain why both a loop bound and a test condition are needed.

Show a row from attempt 2 and explain its mapping. How would you prove that failed practice never reaches test?