SD5913 / WEEK 09 / 5 NOVEMBER 2026

One scene. Two views.

Connected interactions / messages, state and useful failure

Today

01 Locate the other process.

02 Validate an action before changing state.

03 Share one scene through two views.

04 Inspect failure before deciding to retry.

05 Choose a boundary for your project.

What should the screen say when the other program disappears?

OPENING / DATED NEWS

A missing value needs a rule.

FastAPI 0.162.0 / 23 October 2026

Omitted form fields now follow declared defaults or missing-field errors.

Primary release note

What should an omitted gallery control mean: false, unchanged or invalid?

Who starts the tide request?

A person chooses September.

The browser asks the Python service for records.

The response is data; the browser still draws the chart.

01

Find the process.

HOST, PORT, ROUTE AND PROTOCOL

ADDRESS

Four parts of one destination

`http`	`127.0.0.1`	`8009`	`/state`
1 / protocol	2 / host	3 / port	4 / route

One URL: http://127.0.0.1:8009/state

ONE JOURNEY / DIFFERENT JOBS

JSON is not the network.

4 / Application	HTTP / WebSocket / JSON action
3 / Transport	TCP stream / UDP datagrams
2 / Address	IP: which machine?
1 / Link	Wi-Fi / Ethernet / loopback

PREDICT / LOCATE THE FAILURE

Wrong route or no listener?

Server running:

GET /statee

HTTP 404 / application answered.

Server stopped:

GET /state

Connection failure / no HTTP response.

```
Same host.
Same port.
Different failure boundary.

Read the status and body.
A response is not automatically success.
```

Start one Python process.

```
cd week09
uv run python server.py
# Open twice: http://127.0.0.1:8009

# A snapshot in another terminal:
curl http://127.0.0.1:8009/state
```

One process, loopback only. Ctrl+C stops it.

READ / ACTUAL DATA SHAPE

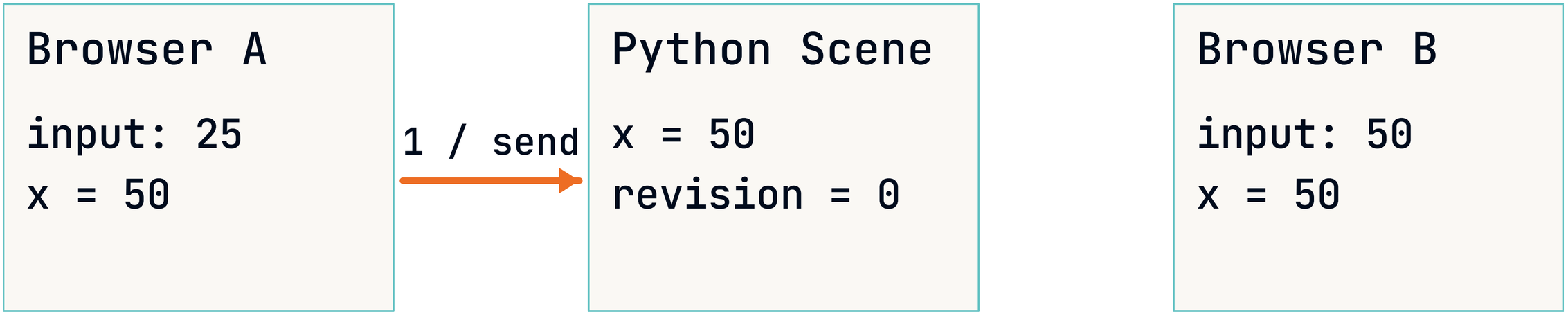
A snapshot describes state.

```
{"type": "state", "x": 50, "revision": 0}

# Send an action:
{"id": "a-1", "action": "set_x", "x": 25}
```

A read and a command have different jobs.

The slider is still local input.




```
{"id":"a-1", "action":"set_x", "x":25}
```

PYTHON / VALIDATION BEFORE MUTATION

Would True be a valid position?

```
if type(message["x"]) is not int or not 0 ⩽ message["x"] ⩽ 100:
    raise ValueError("x must be a whole number from 0 to 100.")
self.x = message["x"]
self.revision += 1
```

Excerpt from Scene.apply; other field checks precede this.

One Scene owns the accepted value.

```
Browser A

input: 25
x = 50
```

```
Python Scene

x = 25
revision = 1
```

```
Browser B

input: 50
x = 50
```

Validate, then change the one authoritative Scene.

An invalid action changes nothing.

x=0 or 100 / accepted

x=101 / rejected

Valid JSON is not enough.

Inspect x and revision after rejection.

```
HTTP 422
{"type":"rejected",
 "id":"a-2",
 "error":"x must be ..."}

Same rule over WebSocket.
```

02

A stream needs boundaries.

A COMPACT RAW TCP OBSERVATION /50-60MINUTES

LIVE / TCP ON 8010

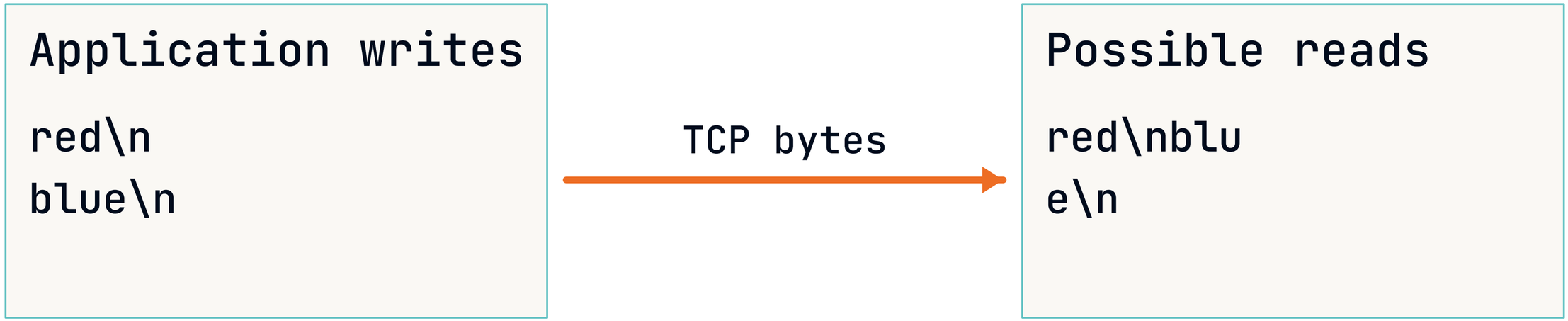
One connection, several messages.

```
# Terminal1, inside week09:
uv run python echo_server.py

# Terminal2:
uv run python echo_client.py
```

TCP newline framing; this is not HTTP or WebSocket.

One send is not one receive.



Keep a buffer; extract a message at each newline.

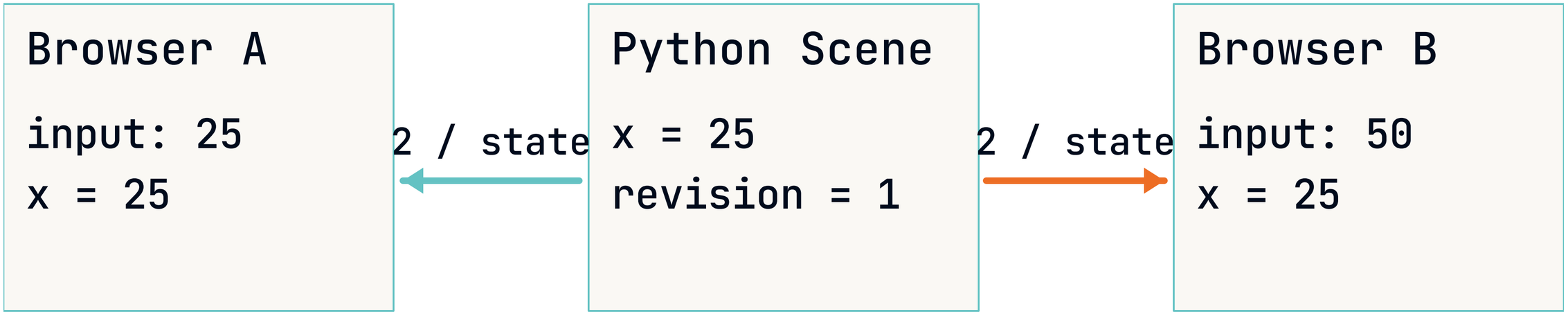
Break / 10 minutes

03

Keep both views current.

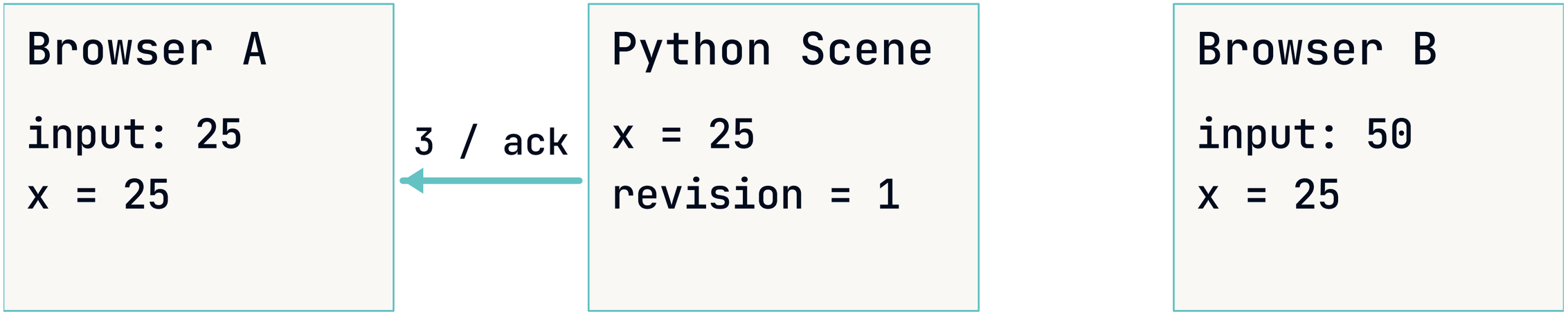
SNAPSHOTS, CONTINUING MESSAGES AND AWAITING WORK

Both views receive the same revision.



{"type":"state", "x":25, "revision":1}

Which action did the result confirm?




```
{"type":"ack", "id":"a-1", "revision":1}
```

Ask again, or receive a push?

HTTP GET /state

One snapshot for this request.

Polling repeats those reads.

HTTP POST /action

One command and response.

```
WebSocket /ws
A continuing bidirectional channel.
State arrives when server broadcasts.

HTTP may reuse its TCP connection.
Persistent transport is not saved data.
```

What can happen during await?

```
incoming = await client.receive()
# Parse and validate this message.
async with lock:
    ack = scene.apply(message)
    await broadcast()
    await client.send_json(ack)
```

Excerpt from updates(); one server process and event loop.

PREDICT / TWO CLIENTS

A sends 25. B sends 80. Which wins?

Inspect the server revisions.

The last accepted action determines x.

A local clock does not establish server order.

A new process starts its revision at0.

04

Read the failure.

95-110 MINUTES / OBSERVED, REJECTED OR UNKNOWN

LIVE / DISCONNECT AND RECONNECT

A new snapshot is not a retry.

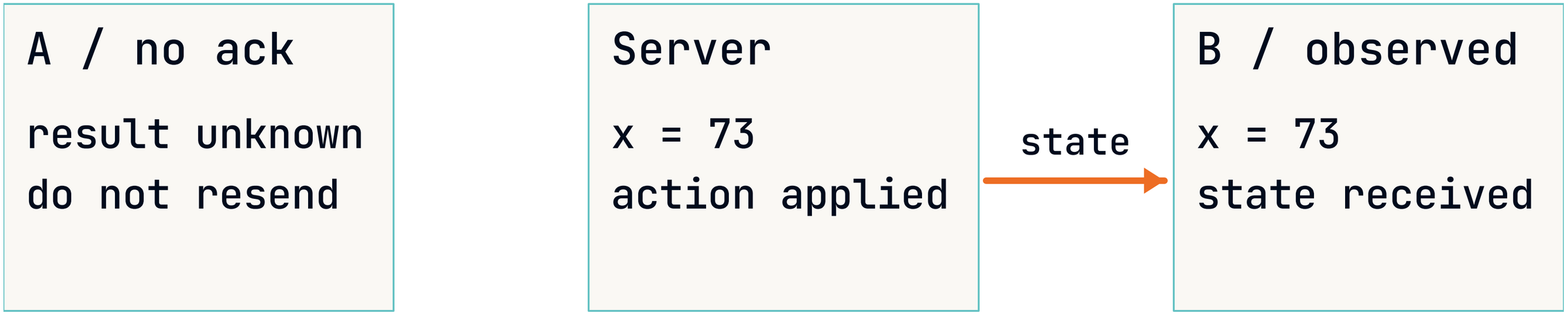
Disconnect A.

Change the scene from B.

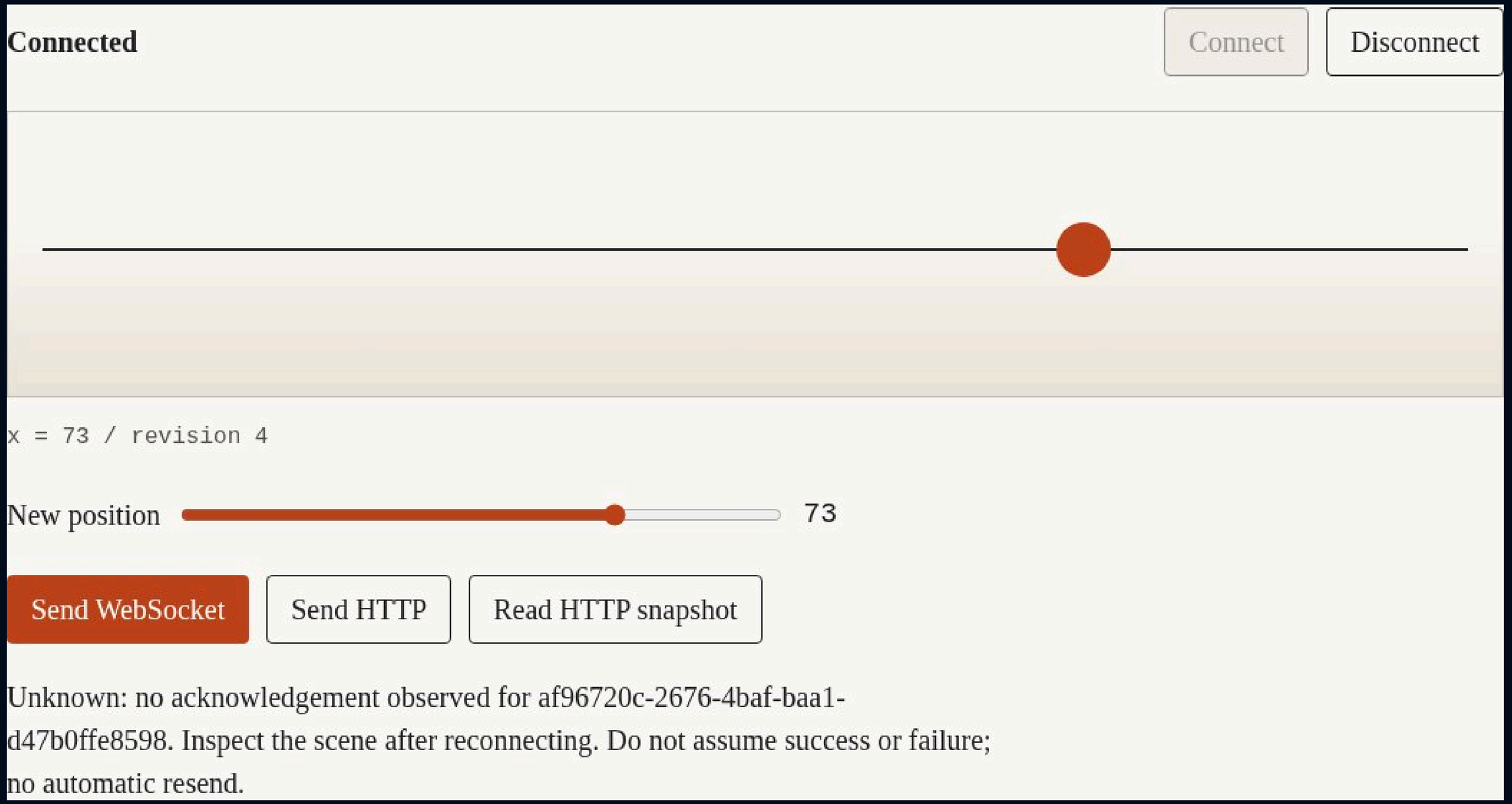
Reconnect A: inspect the current snapshot.

A does not automatically resend its old action.

The sender and observer know different things.



Missing observation does not prove a missing action.



ACTUAL BROWSER / 29 OCTOBER 2026

Connected again; the missing acknowledgement is still unknown.

An action ID is not a duplicate guard.

set_x=73 repeated: same position, another revision.

spend_coin repeated: potentially another cost.

Inspect state before deciding to retry.

This teaching server does not deduplicate IDs.

LIVE / STOP AND RESTART

The process forgot. The browser did not.

Stop Python / disconnected UI.

Last-knownposition stays labeled.

Restart Python / x50, revision0.

Read HTTP snapshot without WS reconnect.

```
Knowledge of old state
is not a current observation.

RAM is not persistent storage.
Week 10 will save related records.
```

What does this interaction need?

EVENT

Spend one coin

Reliable delivery plus application rules.

UPDATE

Move an avatar

Frequent position snapshots;freshness matters.

READ

Fetch a saved record

HTTP request/response is often enough.

CREATIVE BRIDGE / NO EXTRA SETUP

Send coordinates, not camera frames.

Week 8 fingertip x → round(x*100) → set_x.

No hand → no action; clamp to the allowed range.

OSC /scene/x could carry one typed value.

An adapter would translate it to this JSON contract.

TEAM / CHOOSE ONE BOUNDARY

What should another person be able to observe?

Local-only, request/response, or continuing messages?

Name one state, one command and one failure.

Explain what the screen can honestly confirm.

PFAD / SEPARATE 120-MINUTE TUTORIAL

Run it. Change one rule. Prove the boundary.

Pinned Week 9 source and complete tutorial

uv run python server.py

Two views → invalid 101 → disconnect/restart.

Change the maximum from 100 to 80; check both transports.

Final 20 minutes: tutor review / project support.

Trace the message. Design the silence.

Know what changed, what was observed, and what remains unknown.

SD5913.GITHUB.IO/TEACHING

a·t4x