FlowStorm

Conj 2025 workshop

Before we start, let's know each other a little bit:

- Me
  - I'm Juan (coming from Uruguay)
  - Been working as a programmer since the 90s
  - Professionally as a Clojure developer since 2011 on many domains
  - Have a passion for dev tooling and anything that makes writing software more fun
  - Wrote and currently maintain FlowStorm and other OSS
- What about you ?
  - Programming experience
  - Clojure experience
  - FlowStorm experience

Agenda

- A high level FlowStorm overview (what is it?)
- Installation
- Basics
  - Recording & exploring simple functions executions
  - Exploring values with Data windows
  - FlowStorm's REPL API (recordings from the repl)
- Understanding systems Part I
  - A web app
  - Minesweeper clone (ClojureScript)
- Flowbooks
- Understanding systems Part II
  - RadioSnake (a lot of traces)
  - ClojureScript compiler
- Final Q&A

What problem is FlowStorm trying to solve?

- Interactive programming allows us to grow a system iteratively and interactively.
- We can add/change/run functions, but their execution is opaque.
- There is no way of "looking" at execution, other than projecting it via logging/tapping.
- If we can't see, we have to guess a lot, especially in Clojure.

What is FlowStorm?

Editor/IDE independent dev tool, for recording and visualization of Clojure programs :

- execution
- data (like portal, morse, etc, with its own take)

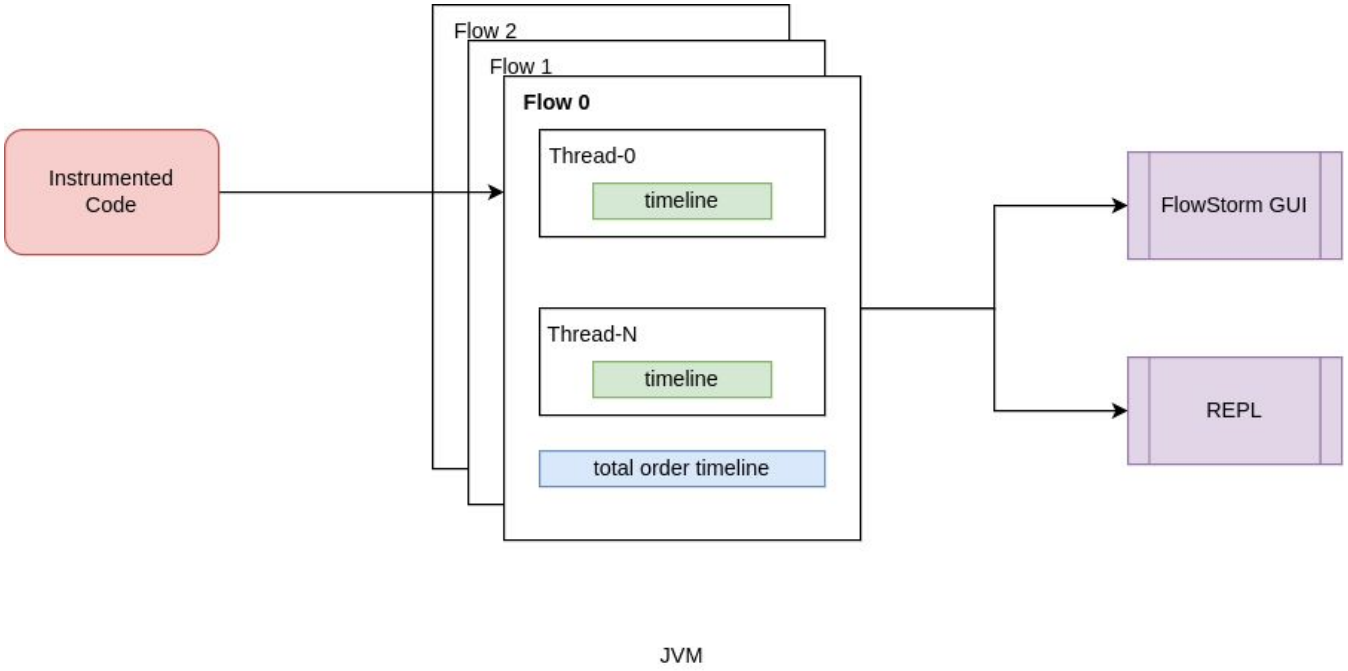
with interactive programming in mind (for dev, a repl companion)

How can it help?

More than a debugger

- Learning/Understanding, from simple functions to entire systems, from a runtime perspective
- With debugging
- With iterative/interactive development by value capture (inline defs, etc)
- Enhancing static code reading by providing more details like values shapes
- Visualizing complex and possibly real time changing data

The big picture (local / in-process)



What do we need for the workshop?

- A recent version of Clojure cli or Leiningen
- JDK >= 21 (for ClojureScript examples, FS requires >= 17)
- NPM (for ClojureScript demo)

```
$ git clone https://github.com/jpmonettas/conj-workshop-2025
```

What’s on the repo?

Global installation

- For clojure cli add conj-workshop-2025/config_samples/deps.edn aliases to
  - ~/.clojure/deps.edn (linux/macos)
  - %USERPROFILE%\.clojure\deps.edn (windows)
- For leiningen add conj-workshop-2025/config_samples/profiles.clj profiles to
  - ~/.lein/profiles.clj (linux/macos)
  - %USERPROFILE%\.lein\profiles.clj (windows)

As you can see there are aliases for FlowStorm itself (:1.12-storm), and aliases for plugins (:fs-*-plugin)

And that’s all we need!

- We can try it by running :

```
$ clj -A:1.12-storm

user=> :dbg
```

- Help -> [Tutorial | User's guide]

Note:

You are going to see some logging. FlowStorm uses JUL so it can be disabled if needed with a logging.properties file.

Basics

- Go to conj-workshop-2025/projects/basics
- Open src/basics/simple.clj
- Start a repl with :1.12-storm (alias or profile)
  - (there is already a .dir-locals.el for Emacs users)
  - (there is a non-needed copy of the alias on deps.edn for VSCode users)
- Load the entire file
- Eval :dbg to start the UI
- When set globally there is no need for any FS stuff on your projects
- Follow the comments in the file

Data Windows

The goals are to provide :

- a way to navigate nested structures in a lazy way
- lazy/infinite sequences navigation
- metadata navigation
- multiple visualizations for each value
- tools for the user to add custom visualizations
- clojure.datafy navigation out of the box
- a mechanism for changing/realtime data visualization
- great integration with the rest of FlowStorm

Data Windows

- Open basics/src/data_windows_demo.clj
- Follow the comments in the file

REPL API

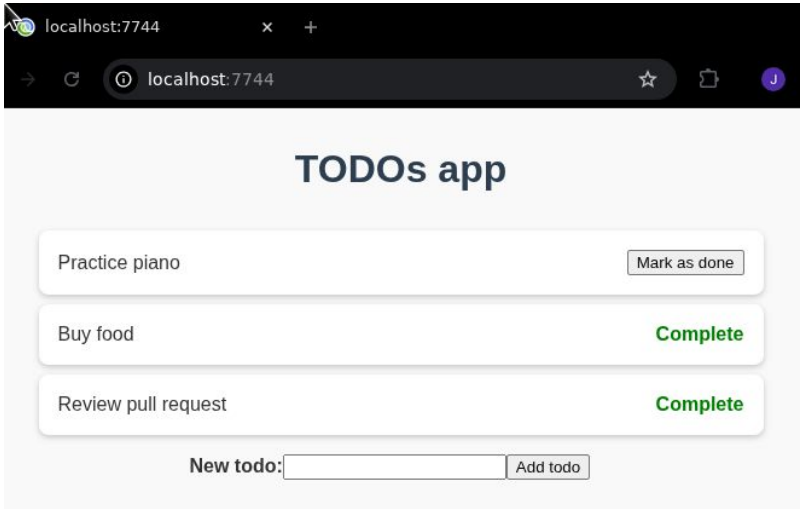
The REPL api allows us to work with recordings from the repl, skipping the UI.

This can be used in many ways, from finding patterns programmatically and custom visualizations (the plugin system) to being a interface for LLMs to use FlowStorm.

- Open src/basics/programmable.clj

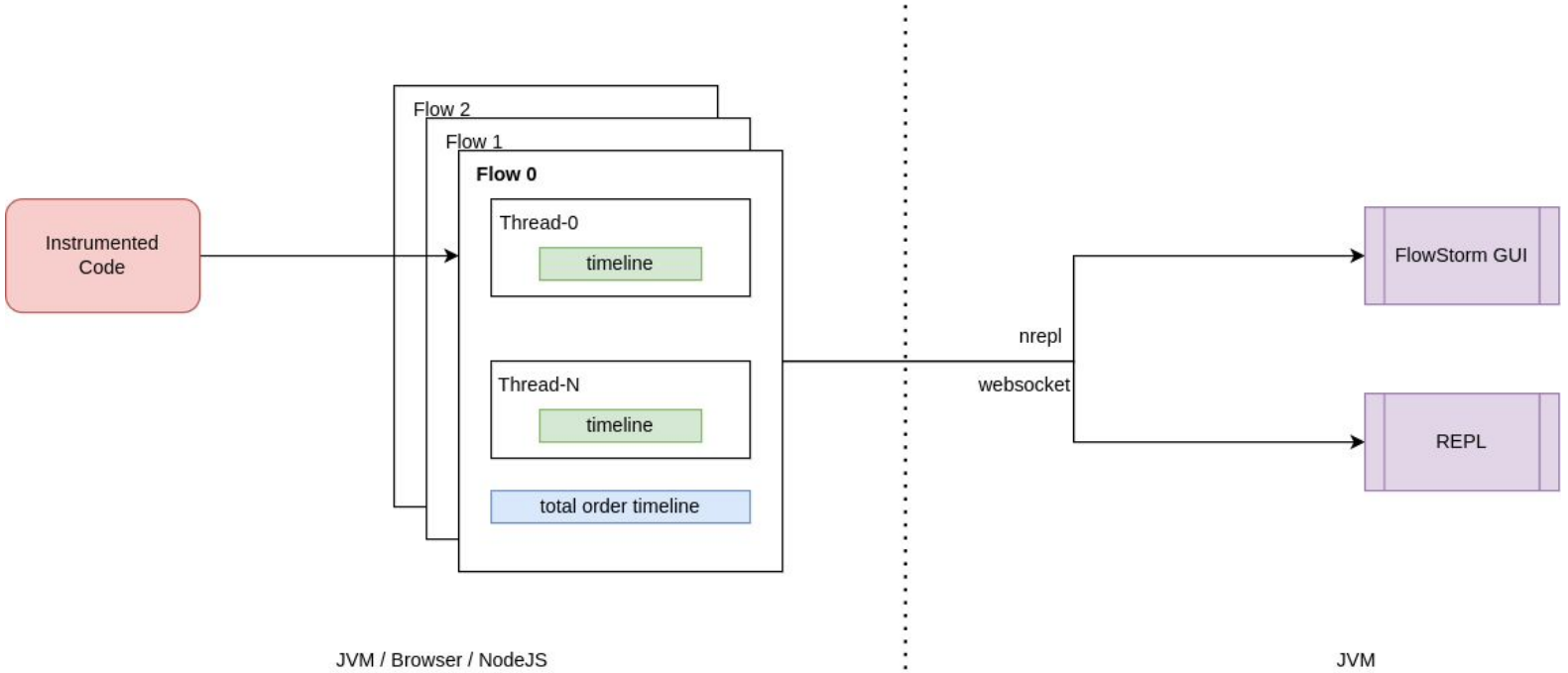
Understanding systems (Web App)

- Go to conj-workshop-2025/projects/web-app
- Open web-app/src/todos/web_app.clj
- Start a repl with :1.12-storm:fs-web-plugin (aliases or profiles)
- Follow the comments at the bottom of the file



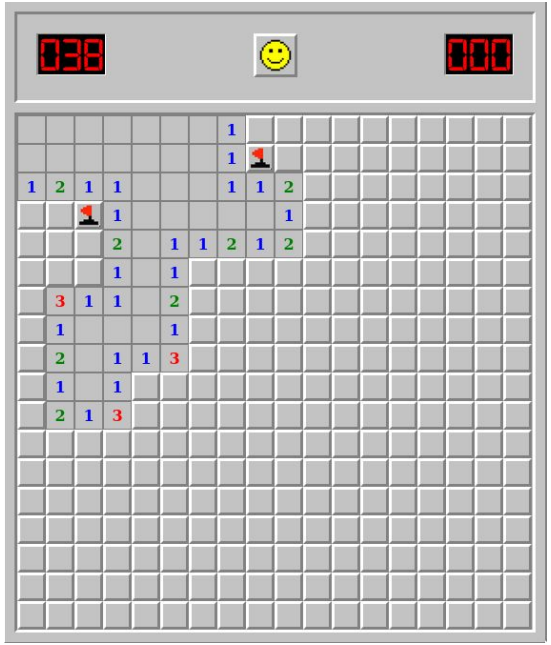
Break

The big picture (remote)



Understanding systems (Buggy minesweeper)

- Go to conj-workshop-2025/projects/minesweeper
- Look at shadow-cljs.edn
- Open minesweeper /src/mine_sweeper/core.cljc
- Run :
  - $ npm install
  - [term1] $ npx shadow-cljs watch :dev (watch/compile)
  - [term2] $ ./run-debugger[.bat] (FS GUI)
  - [term3] $ npx shadow-cljs cljs-repl :dev (cljs repl)
  - Open http://localhost:8021
- Follow the comments at the bottom of the file

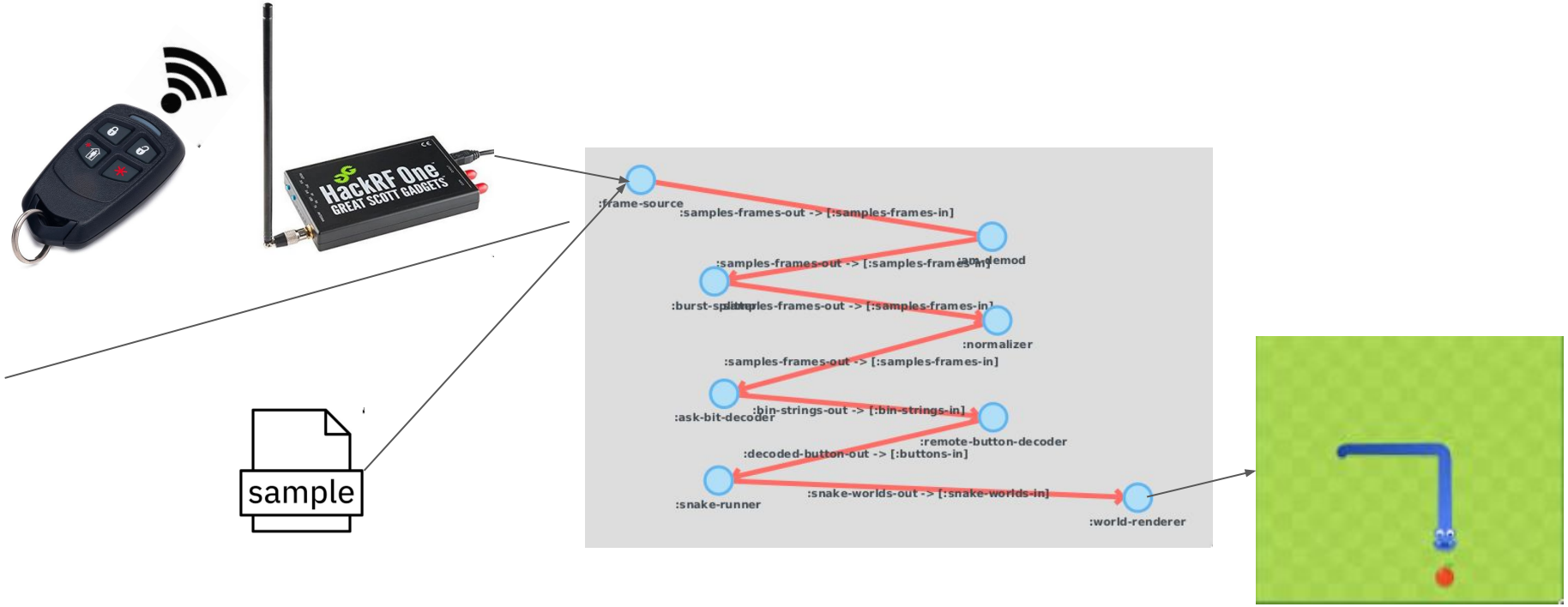


Flowbooks

The flowbook plugin allows you to store and replay your flows with an optional "flowbook", which is a notebook that supports links to your recordings you can use to explain a flow.

- On a terminal to `conj-workshop-2025/projects/flowbooks`
- Run `clj -A:1.12-storm:fs-flowbook-plugin`
- Run the ui `(:dbg)`
- Go to Flowbooks, load `flowbooks/todos-web-app/todos-web-app.edn`
- After the flows are loaded, open the “flow book”
- Edit and modify `todos-web-app-flowbook.html` then refresh

Understanding systems (RadioSnake Demo)



Understanding systems (RadioSnake Demo)

- Go to `conj-workshop-2025/projects/radio-snake`
- Open `radio-snake/dev/dev.clj`
- Take a look at `radio-snake/deps.edn`
- Start a repl with `:dev:1.12-storm:fs-async-flow-plugin`
- Load `dev.clj`
- Run `:dbg`
- Follow the comments at the bottom of dev.clj

Understanding systems (ClojureScript compiler Demo)

- Goto `conj-workshop-2025/projects/cljs_compiler`
- Open `src/user.clj`
- Start a repl with `:1.12-storm:fs-cljs-compiler-plugin`
- On a terminal on the same folder run:
  - `$ lein repl :connect localhost:`cat .nrepl-port``
  - `user=> (cljs.main/-main "--repl")`
- Record the compilation of something like `(defn my-sum [a b] (+ a b))`
- Run the plugin

Notes on instrumentation (Vanilla vs ClojureStorm)

There are two ways of using *FlowStorm* for Clojure :

- With ClojureStorm (recommended) : Swap your Clojure compiler at dev time by ClojureStorm and get everything instrumented automatically.

- Vanilla FlowStorm : Instrument by tagging and re-evaluating forms (like cider debugger).

Resources

- http://www.flow-strom.org
- https://flow-storm.github.io/flow-storm-debugger/user_guide.html
- https://github.com/flow-storm/flow-storm-debugger

Thank you!

Final: Q&A