Fellowes

Writing Object Oriented Elixir Architecture

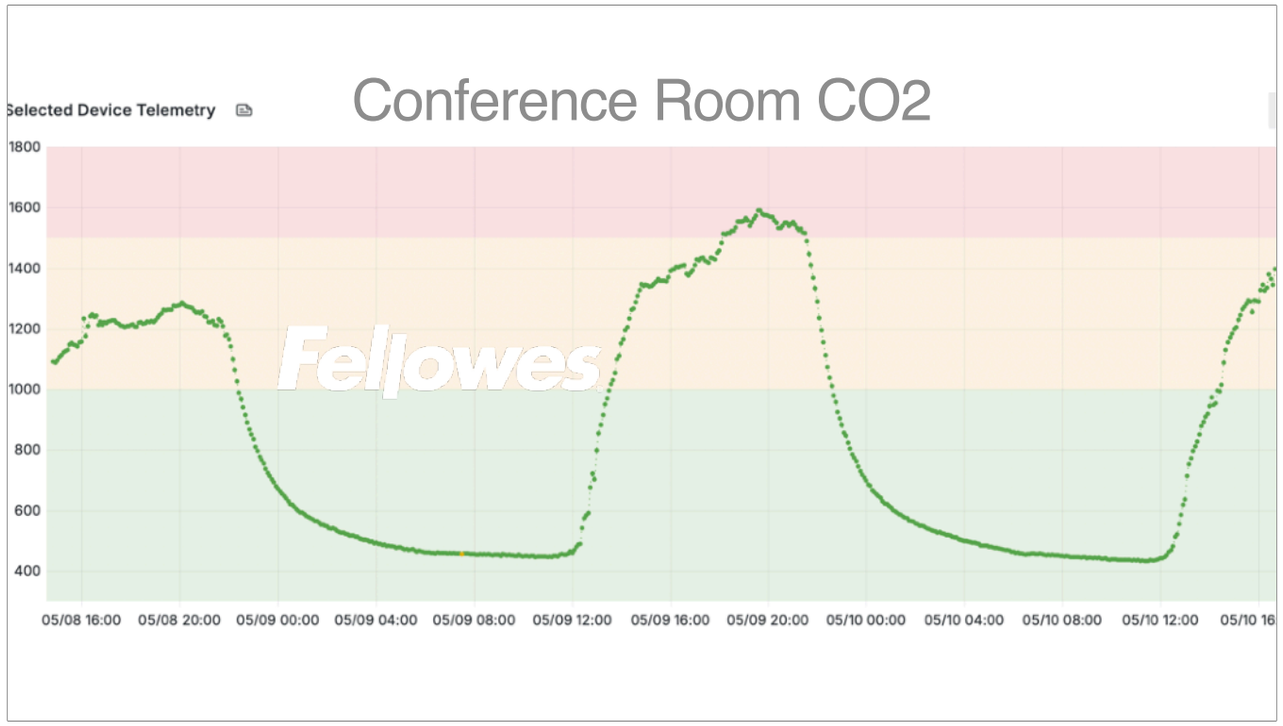


Taun Chapman

Fellowes Staff Engineer

Goal:

Motivate you to continue to create improved tools for software development in the Elixir ecosystem by giving you as yet unrealized concrete examples from the past.



These are actual CO2 readings in the conference space during the conference. No wonder I felt like falling asleep during the afternoon talks! It wasn't lunch. It was the very high level of CO2 in the room. "At 1,600 ppm, you may experience symptoms such as headaches, drowsiness, and decreased concentration." Do you know the CO2 in your office, home office, or meeting rooms? Are you dringking more coffee because you have less oxygen available?

We are already OO Programming in Elixir
when we use best practices.

Anyone can write bad code.
There is no need for me to give that talk.

In my opinion Erlang/Elixir/Pony etc are the only languages which can be called OO.

Joe Armstrong

Yes, Erlang is in many ways much more like the original Smalltalk, and Java (and C++) are much further away from the idea.

I thought of objects being like biological cells and/or individual computers on a network, only able to communicate with messages.

Alan Kay

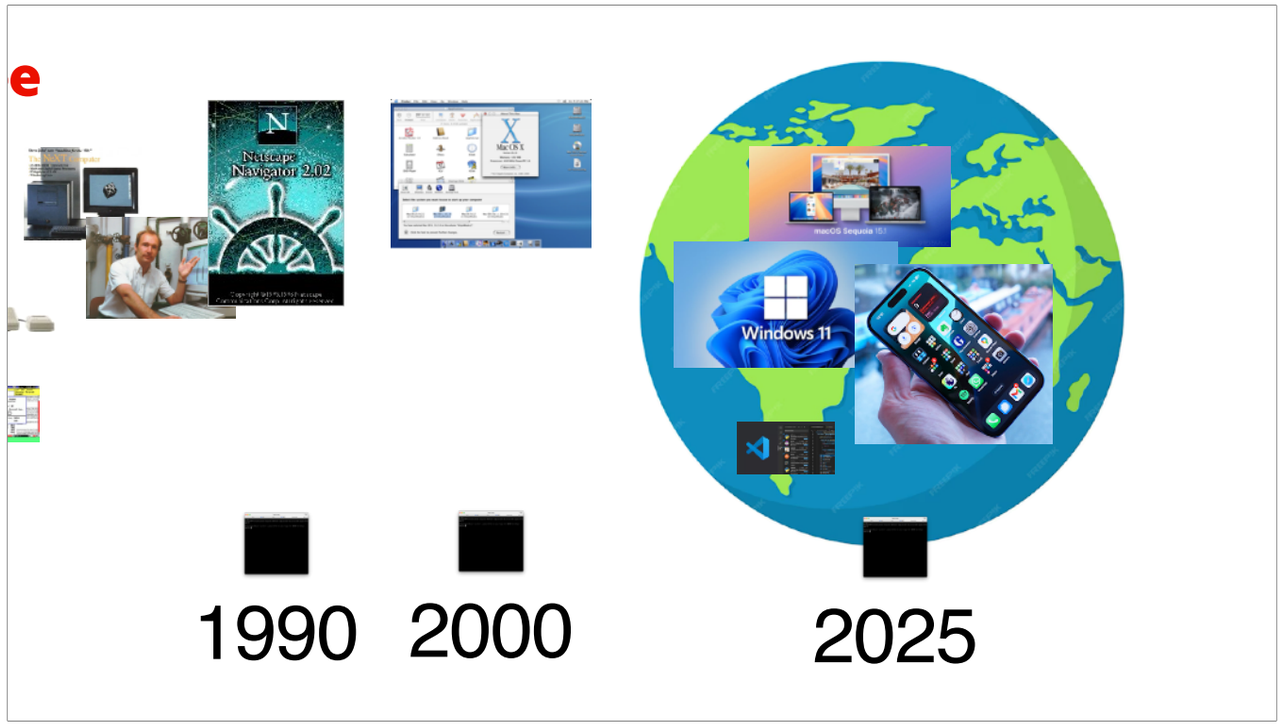
Joe Armstrong and Alan Kay both agree, Erlang/Elixir is Object Oriented. But does the tooling take advantage of that like Smalltalk's did? Does the tooling take the cognitive load off the developer?



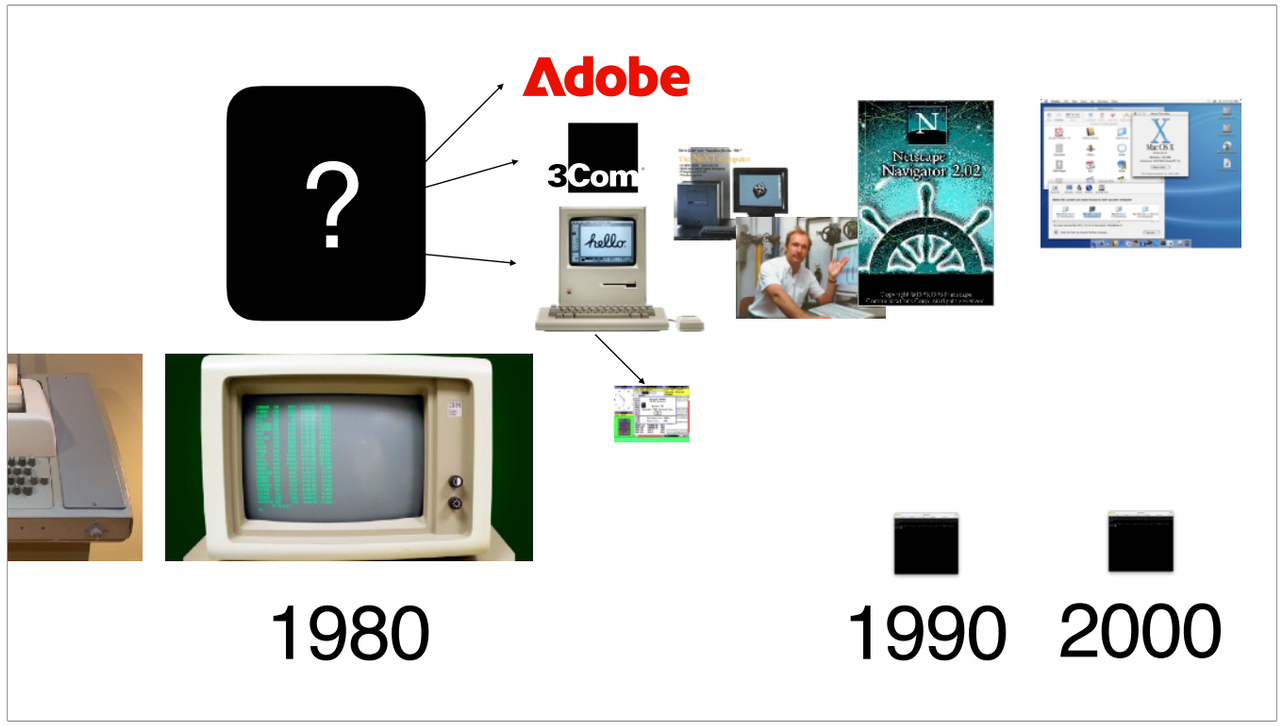
2025

Who runs the world?

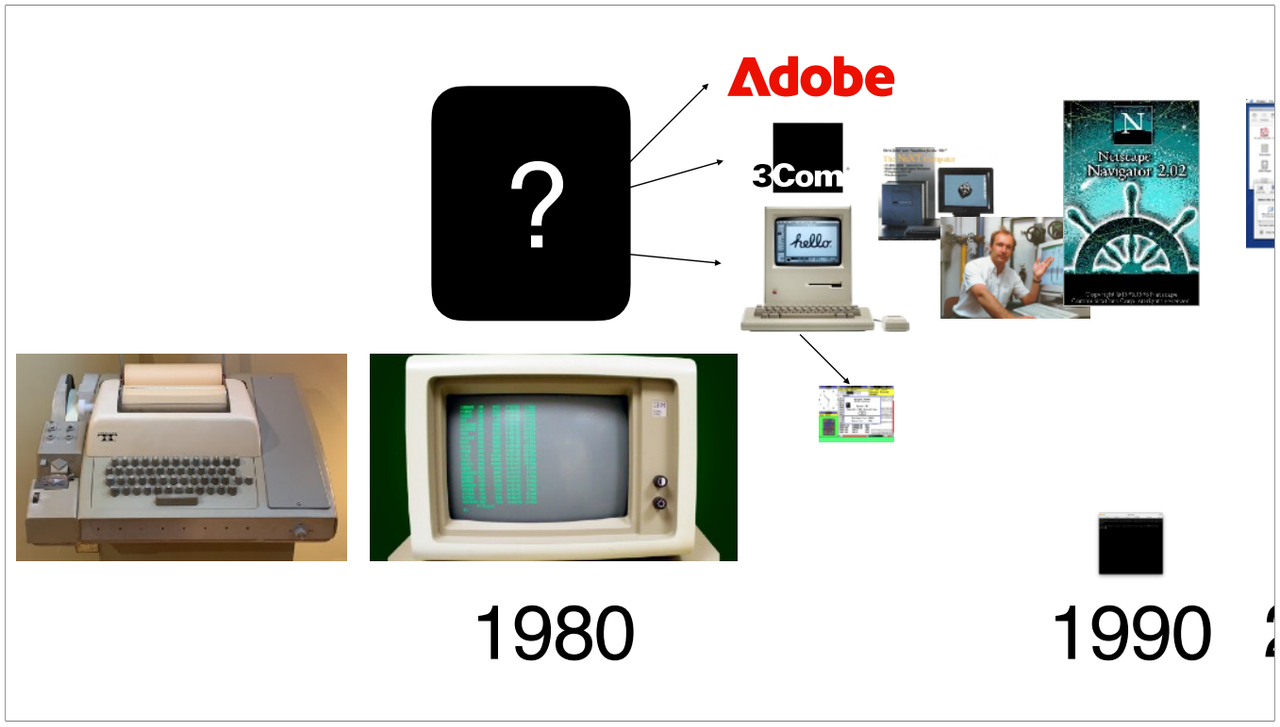
How did we get here? All the GUIs? iPhones running iOS. Windows, Android, … Everything connected by the web.

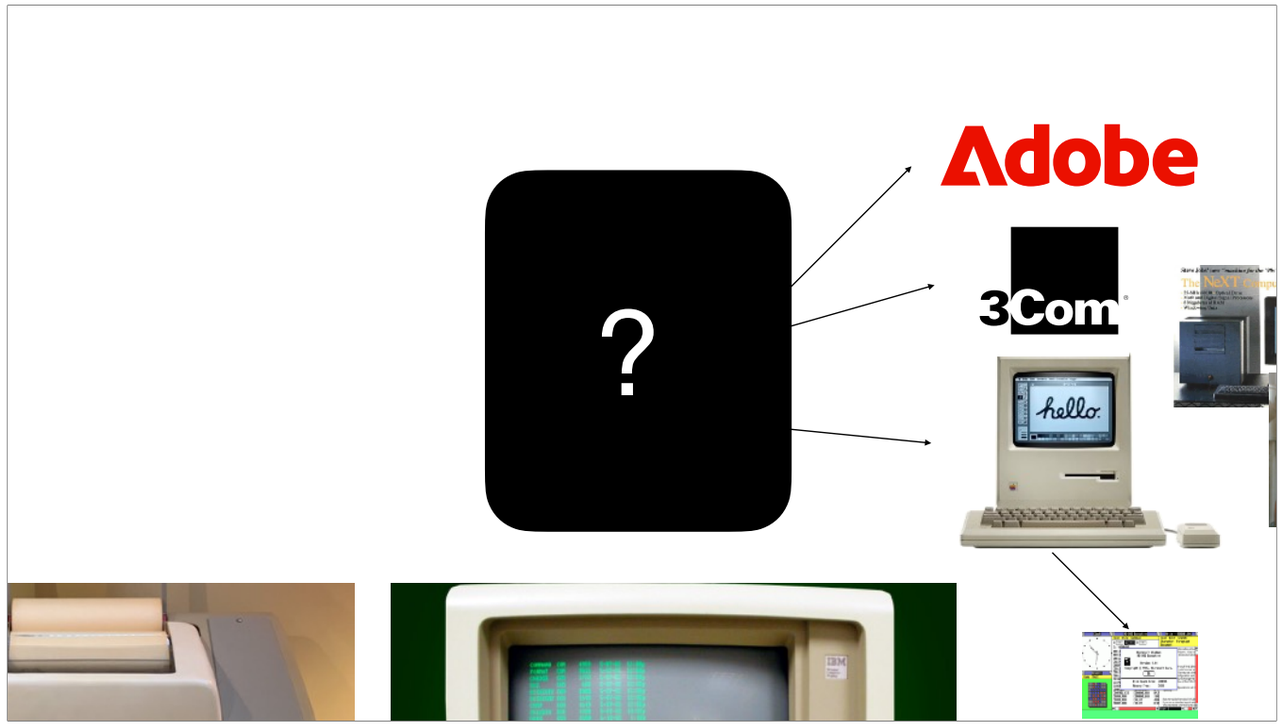


The web, html, and the browser, were invented by Tim Berners-Lee using a NeXT computer with NeXTStep OS. Which became our current iOS and MacOS. It had great tooling for developing software.
Java was meant to be a cross platform OOP but had terrible tooling. But we still got Android.



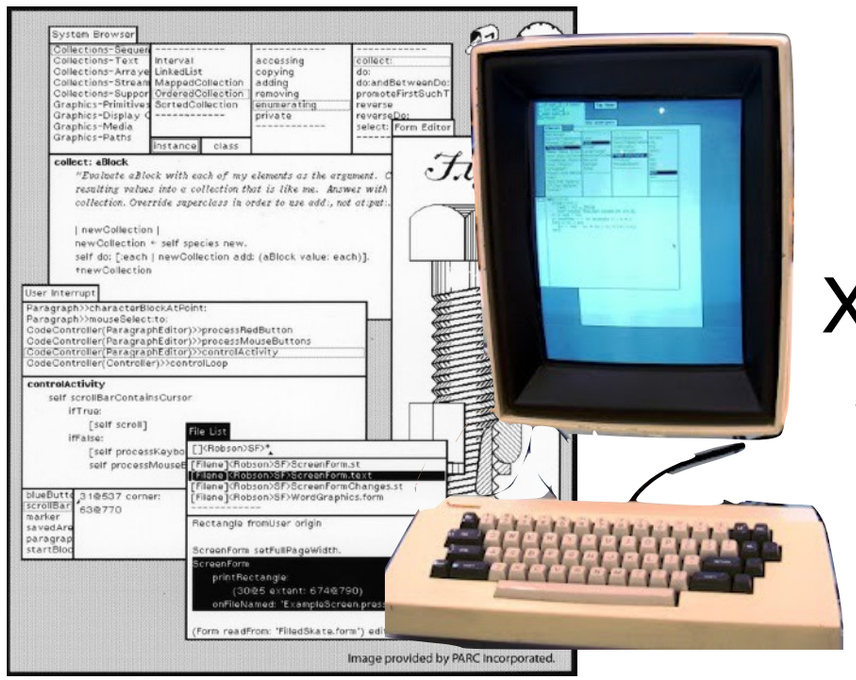
While the rest of the world had teletypes, glowing CRTs, and punched cars, there was on place where you could play on a computer with a graphical menu & mouse driven user interface with a language and tooling so simple, even a child could create with it. They also happen to have invented the internet to connect the computers together.


Adobe
3Com
hello
Netscape
Navigator 2.02
1980
1990



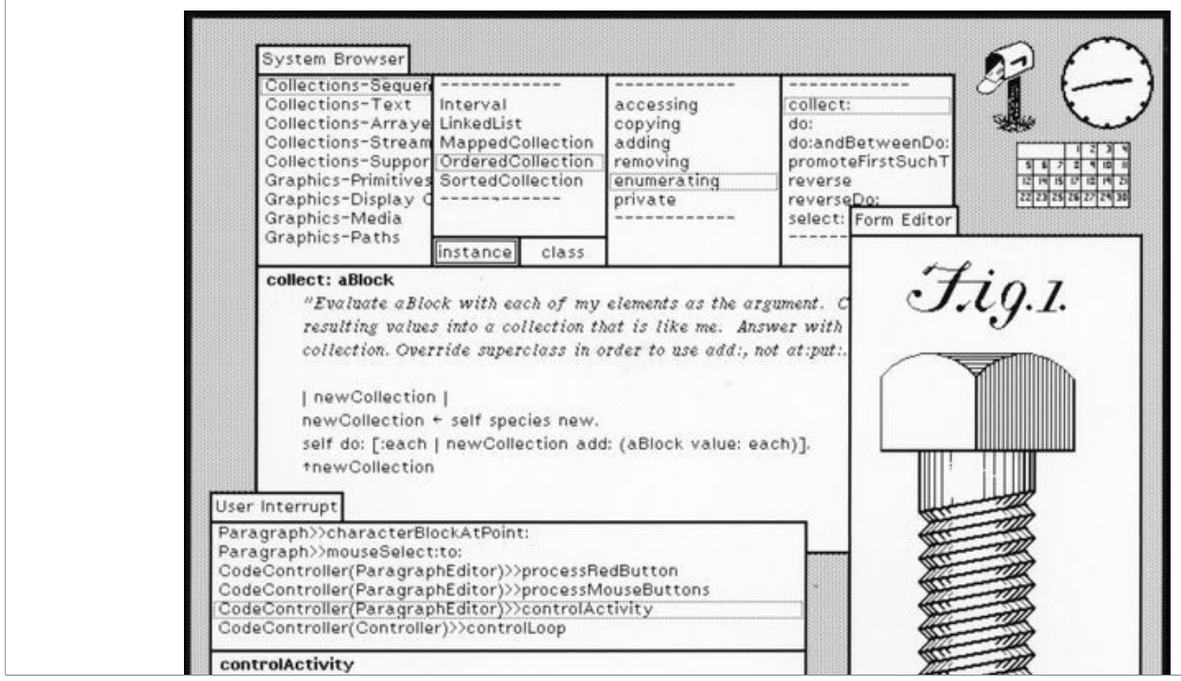
HP, Tektronix, and Apple were invited to a demo. A demo which led to the Mac.

1980



Xerox Alto

That place was Xerox’s Parc Place.
The computer was the Xerox Alto.
The language/OS/Tooling was Smalltalk.
IT was Smalltalk top to bottom. Not just a language. A way of thinking. A way of working.



This is the Smalltalk integrated development environment written in Smalltalk as was all the user interface of the Xerox Alto.
The system browser is a window into the whole OS!! No files. Just classes of functionality organized any way you want with drill down functionality to individual modules, functions within the modules, ways of organizing the functions, like sorting and filter, then the function editor which micro compiles the code in the window when saved. Code can be compiled and run anywhere in the user interface.

Demo



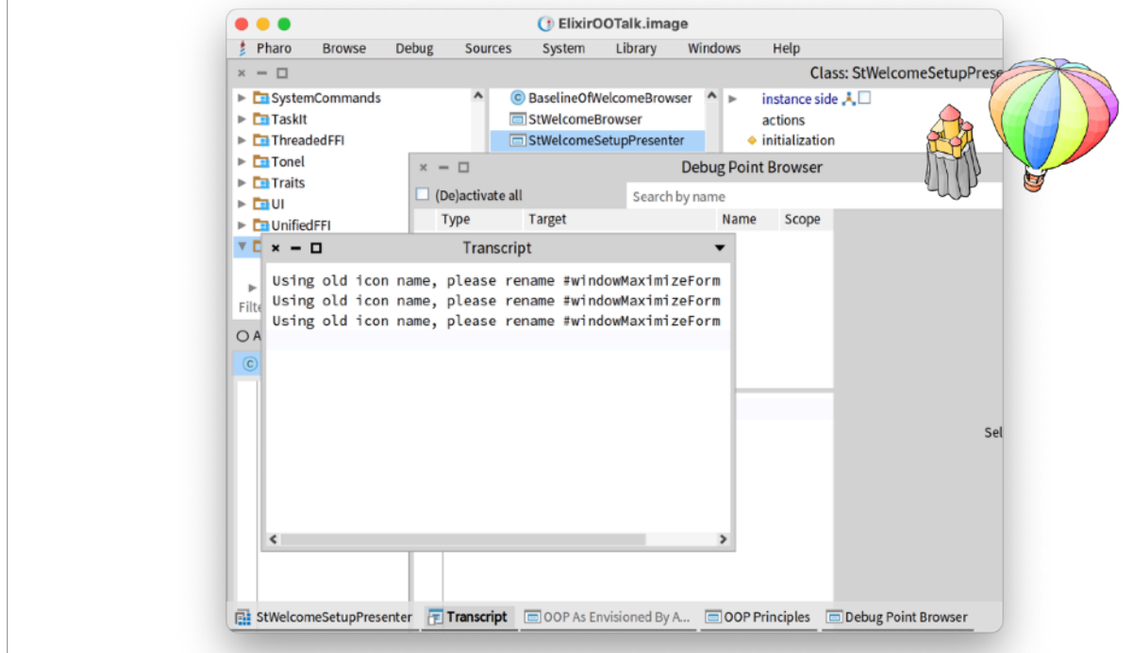
Hands on demo featuring Pharo. The modern Smalltalk. https://www.pharo.org

Smalltalk visibility demo

- Original IDE? Editor helps with organization and encapsulation. Can run code within the editor. **Every editor window is an iex console**. Includes documentation view.
- Inspector! Like all of our console command line introspection or Observer.
- Live debugger - Ability to call any object message while running
- Dynamic - Live changes
- No waiting for compilation
- Consistently won timed programming contests

Demo

- The environment IS Smalltalk
- Every pane is a potential code editor+compiler+linker+hotswap
- Code viewing is a virtual view - not limited by the file layout.
- Debugger - Rollback to previous stack position



There is no relation between code and it’s location in a file.
The Tool, manages the organization for you. Much better than VSCode does.
Code is compiled in the editor.

Inspirations from Smalltalk

- Code as prose - be willing to spell it out.
- Code that is just there for me - visibility functions, helper functions aka tools
- Use “setters” - easier code completion, easier debugging, more visibility
- Code as action - “increment_user_count/2” vs “cnt = cnt++1”
- Keep it small stupid - all the relevant code should always be visible.
  - No don’t just get a bigger screen.
- Decomposition & Encapsulation - Don’t be afraid to break big problems down into smaller problems and treat them as a building block.
- We could use more tools that improve our observability, and iterative speed

Elixir

How OOP principles and Erlang support visibility

- Erlang is pretty unique in having a very similar feature set.
- Encapsulation is like a test point
- Decomposition - improves testability. Testing is a way of observing.
- Introspection - Observer, children, state
- Dynamic - pasting code in console, hot updates
- LiveBook - a web based “playground”

But …

- We still use dbg(), IO.inspect, etc ~ printf
- We still constantly context switch between writing, running, evaluating, writing, …
- We are still managing our text files and saving syntactically broken code.

How Can We …

Take inspiration from Smalltalk
To make our tools more user friendly

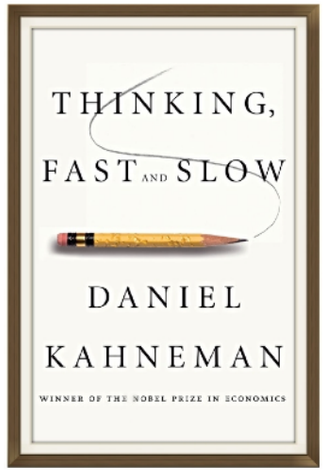
- A view which both renders the stack or a sequence of calls but **also** makes it trivial to use the view to move up and down the call chain
    - A tool to show inferred call graphs? Pick a start point, …
    - Mix deps.tree in LiveBook? Phoenix with links + expand/ collapse node?
- A truly simplified view of Action->A->B->C->D as a hyperlinked list view with a pane for the code but just the function? That is editable? Maybe even a view that shows each function in an editor all at once as if they were in the same file? (**Give an example image from Fellowes code**). Like Trace but truly for Elixir with a Phoenix UI?
- Message histories? Maybe a library which can be plugged into a GenServer message box? Like Erlang et()? Example?

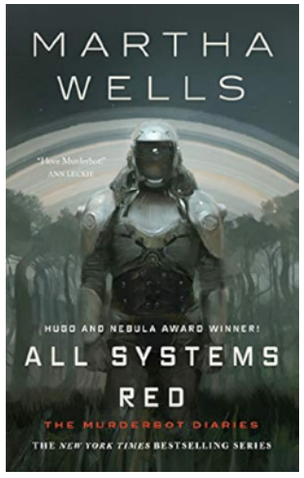
… make what we do easier?

- What if unit_test tests could be like Smalltalk debugger. Pause, edit, save, retry test. Low hanging? Smalltalk inspector is basically a GUI stack inspector with a code editor/compiler, etc. Like LiveBook but more.
- An observer that works? No Wx issues. Maybe via Phoenix like the PhoenixDashboard but more. And shows the last message rather than no messages because the queue is empty?
- LiveBook + Trace + Editor + File ingester <- too much?

- Your ideas here …

Taun's Books of 2024





Thank You

Fellowes