ContextTracking.jl

This is a ***production-quality*** package for keeping tracking of contextual information. The context data is accessible from all functions called from the current execution context, regardless of how deep the nested calls are. The library is thread-safe and usable from async processes.

The problem

The program contains 4 functions such that

- A calls B
- B calls C
- C calls D

Some data were gathered in A and they need to be available in D, ***but maybe not*** in B or C.

The old way...

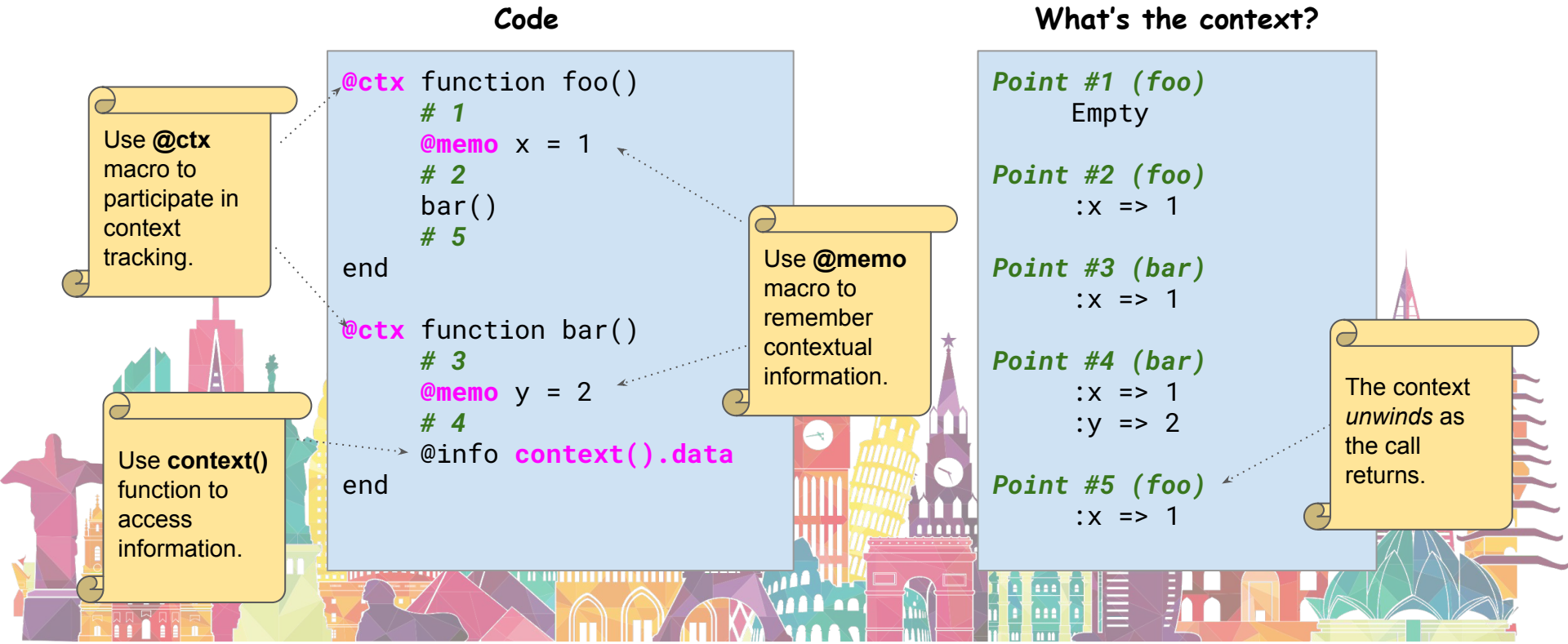
1. Define a Context data type.
2. Create the context object in function A.
3. Add a new context argument in function B & C & D.
4. Pass the object down the chain from A to B to C to D.

ContextTracking!

1. Annotate the functions A, B, C, and D with **@ctx** macro.
2. Record context data using **@memo** macro in A.
3. Access data in function D using **context** function.

Tom Kwong (GitHub: tk3369, Twitter: tomkwong)

What happens during execution?



Tom Kwong (GitHub: tk3369, Twitter: tomkwong)