



The What and Why of Software-Transactional Memory



Let's start with WHY

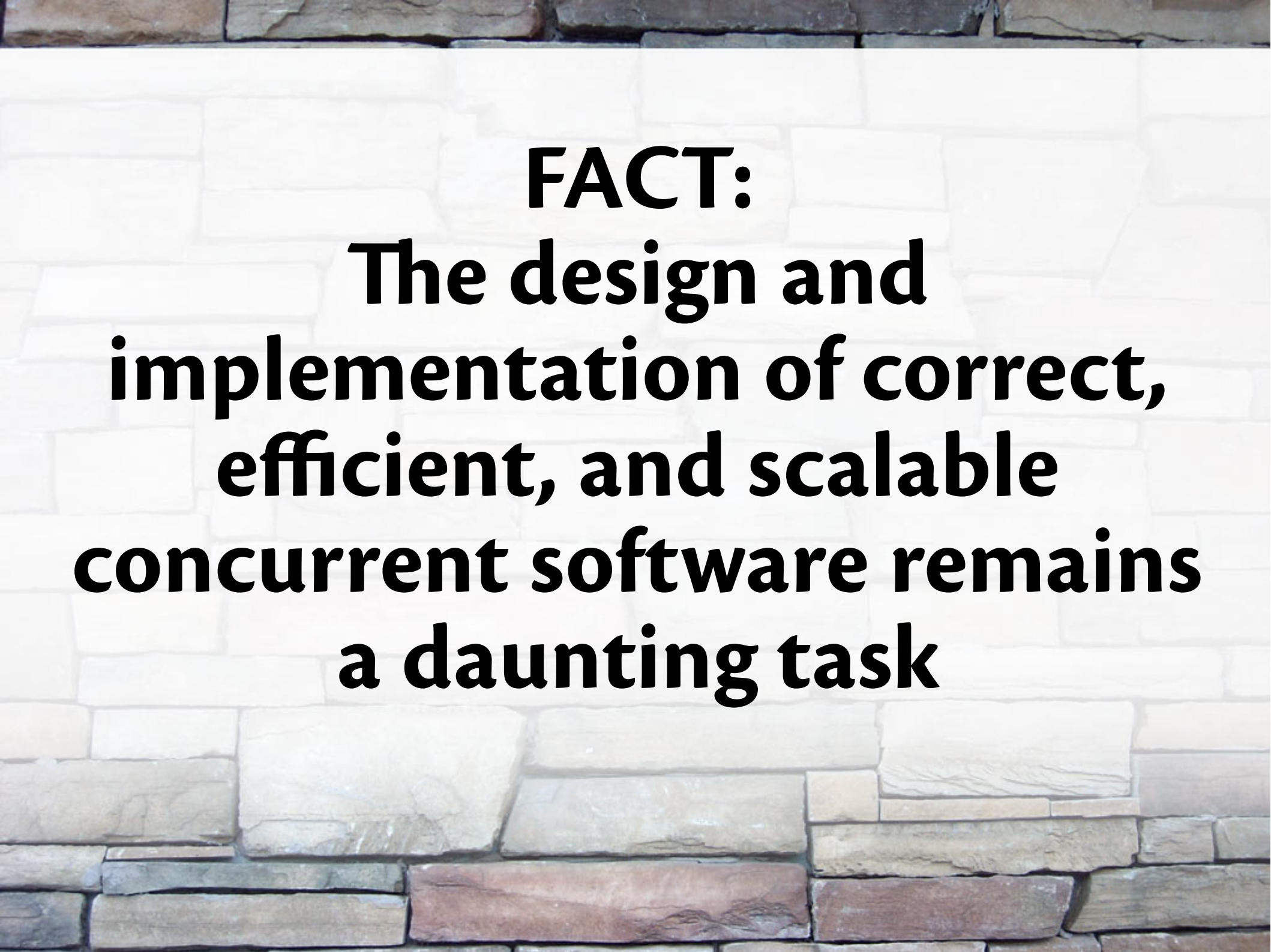
A background image of a stone wall with light-colored rectangular stones in the upper half and darker, more irregular stones in the lower half.

FACT:

**Many modern applications
have increasingly stringent
concurrency requirements**

A background image of a stone wall with light-colored rectangular stones in the center and darker, more irregular stones at the top and bottom.

FACT:
**Commodity multicore
systems are increasingly
affordable and available**



FACT:
**The design and
implementation of correct,
efficient, and scalable
concurrent software remains
a daunting task**

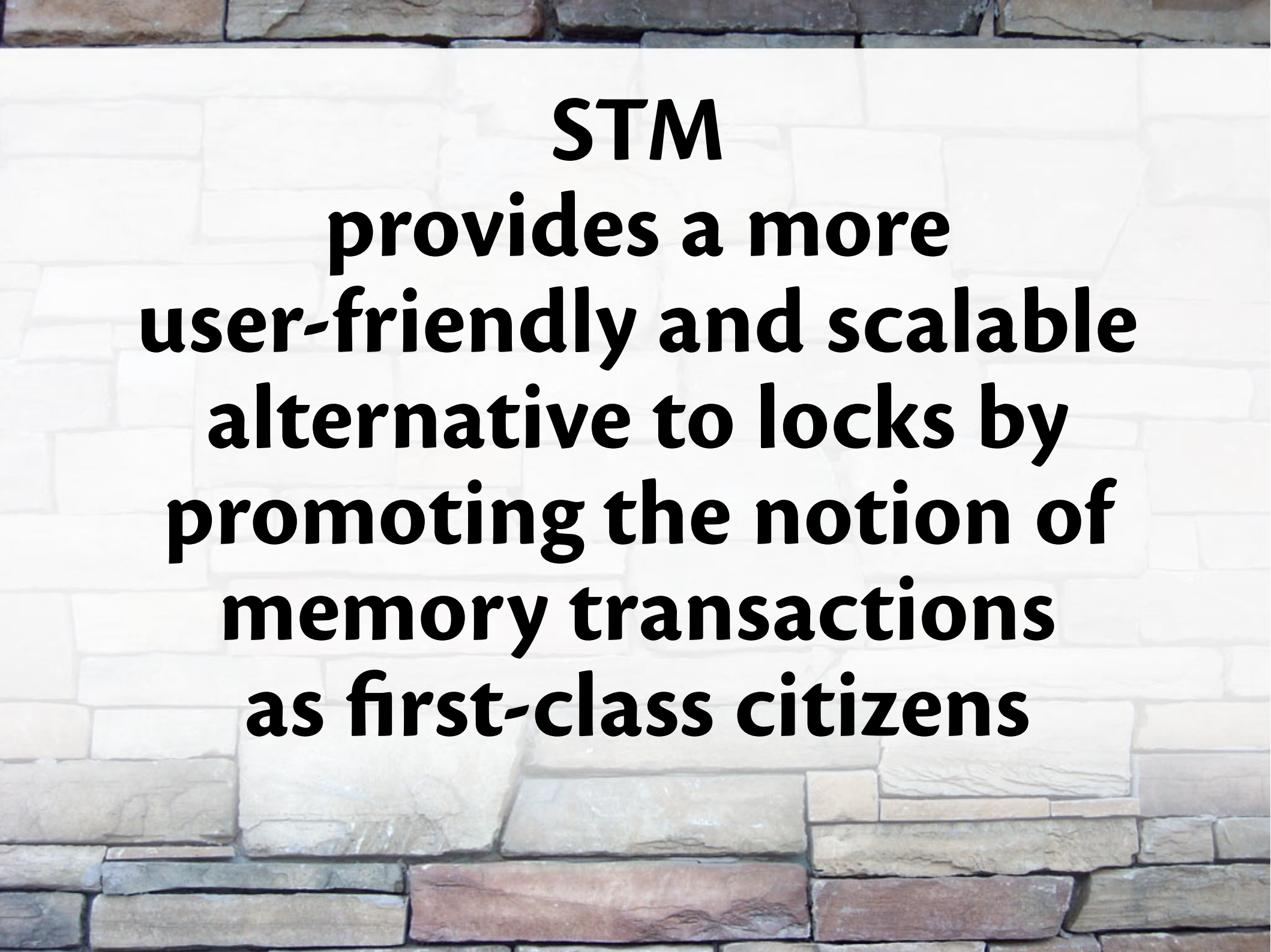
A background image of a stone wall with light-colored rectangular stones in the center and darker, more irregular stones at the top and bottom.

Haskell to the rescue!

Meet STM

The background of the slide is a close-up photograph of a stone wall. The stones are of various sizes and shapes, with colors ranging from light beige to dark brown. The texture is rough and natural. The text is centered over the middle of the image.


STM
protects shared state in
concurrent programs



**STM
provides a more
user-friendly and scalable
alternative to locks by
promoting the notion of
memory transactions
as first-class citizens**

The background of the image is a close-up of a stone wall. The stones are of various sizes and shapes, with colors ranging from light beige to dark brown. A large, white rectangular area is superimposed over the center of the wall, serving as a backdrop for the text.

**Transactions, like many of
the best ideas in computer
science, originated in the
data engineering world**

A background image of a stone wall. The top section features a row of dark, irregular stones. Below this, the wall is composed of larger, light-colored, rectangular stones arranged in a regular pattern. The bottom section returns to a row of dark, irregular stones, mirroring the top. The text is centered in the middle section.

**Transactions
are one of the foundations of
database technology**

**Full-fledged transactions
are defined by the
ACID properties
Memory transactions use
two of them (A+I)**

The background of the slide is a close-up photograph of a stone wall. The stones are irregular in shape and size, with a mix of light beige, tan, and reddish-brown hues. The texture is rough and natural.

**Transactions provide
atomicity and isolation
guarantees**

(SQLite transactions demo)



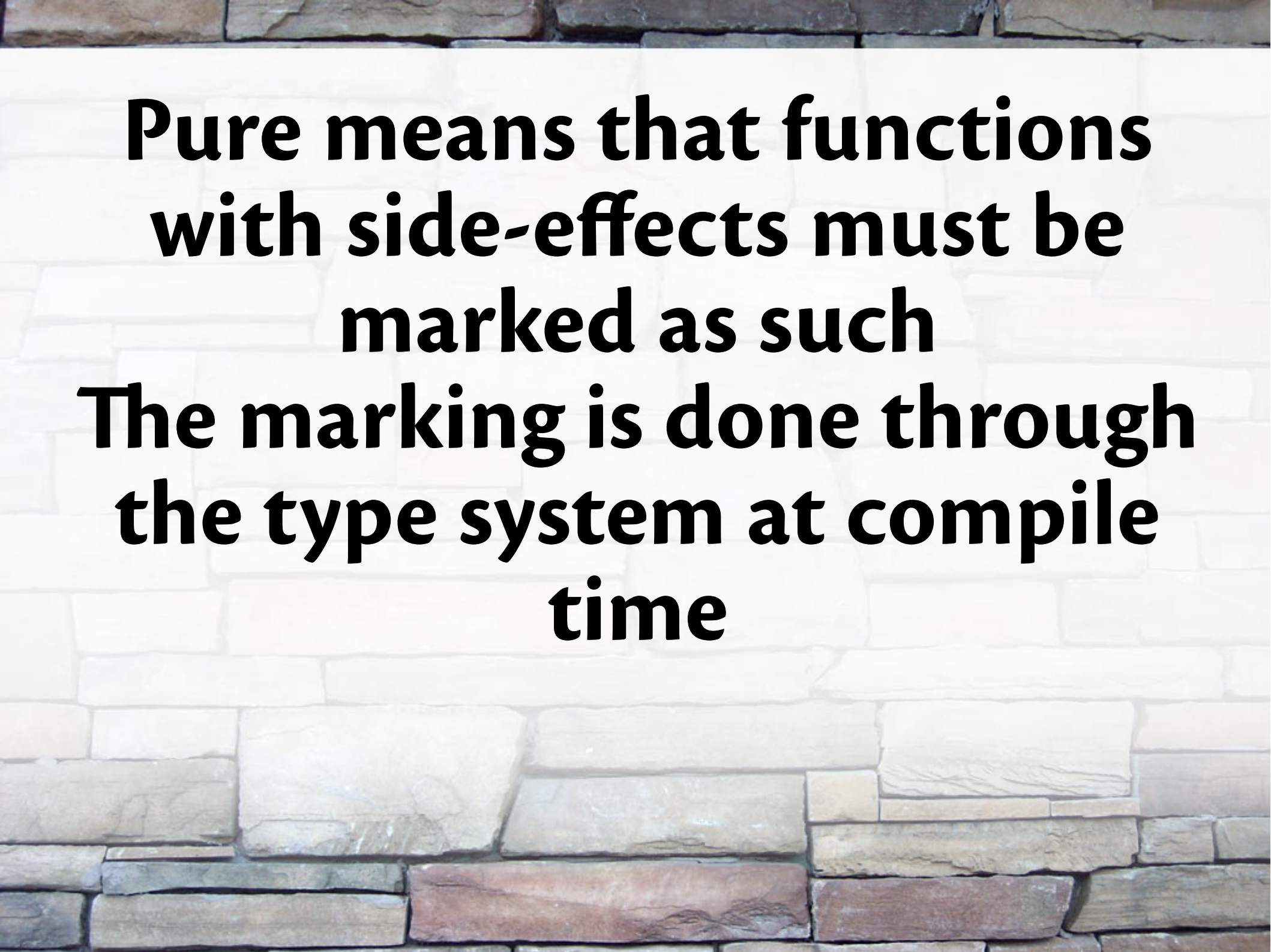
**Strong atomicity
means all-or-nothing**

A background image of a stone wall. The top section features a horizontal band of dark, irregular stones. Below this, the wall is composed of larger, light-colored, rectangular stones with visible mortar lines. The bottom section returns to a band of darker, more varied stones.

**Strong isolation
means freedom from
interference by other
threads**



**Recall that Haskell is a
strictly-typed, lazy, pure
functional language**



**Pure means that functions
with side-effects must be
marked as such
The marking is done through
the type system at compile
time**

STM
is just another kind of
I/O
(with a different marker:
"STM a" instead of "IO a")

A background image of a stone wall with light-colored rectangular stones in the center and darker, more irregular stones at the top and bottom.

**Transactional memory needs
to be declared explicitly as
TVar**

A background image of a stone wall with light-colored rectangular stones in the center and darker, more irregular stones at the top and bottom.


**The STM library provides an
STM-to-IO converter called
"atomically"**

Transactional memory can only be accessed through dedicated functions like "modifyTVar", "readTVar", "writeTVar" which can only be called inside STM blocks

The background of the slide is a close-up photograph of a stone wall. The wall is composed of irregularly shaped stones in various shades of beige, tan, and light brown. The stones are laid in a traditional pattern, with some larger flat stones and some smaller, more angular pieces. The lighting is even, highlighting the textures and colors of the stone.

**Another useful STM
abstraction is the TChan,
an unbounded FIFO channel**

**Once some messages are
transferred into a TChan,
they are ready to be
consumed by other threads
(broadcasting is possible too)**

A background image of a stone wall with light-colored, irregularly shaped stones in the center and darker, more uniform stones at the top and bottom.

TChans are useful when threads need to send signals to each other, as opposed to just accessing shared state

A background image of a stone wall with light-colored rectangular stones in the center and darker, irregular stones at the top and bottom.

Compile your STM code with:

```
ghc -threaded program.hs
```

When running the program:

```
./program +RTS -N
```



Follow me on GitHub

github.com/dserban