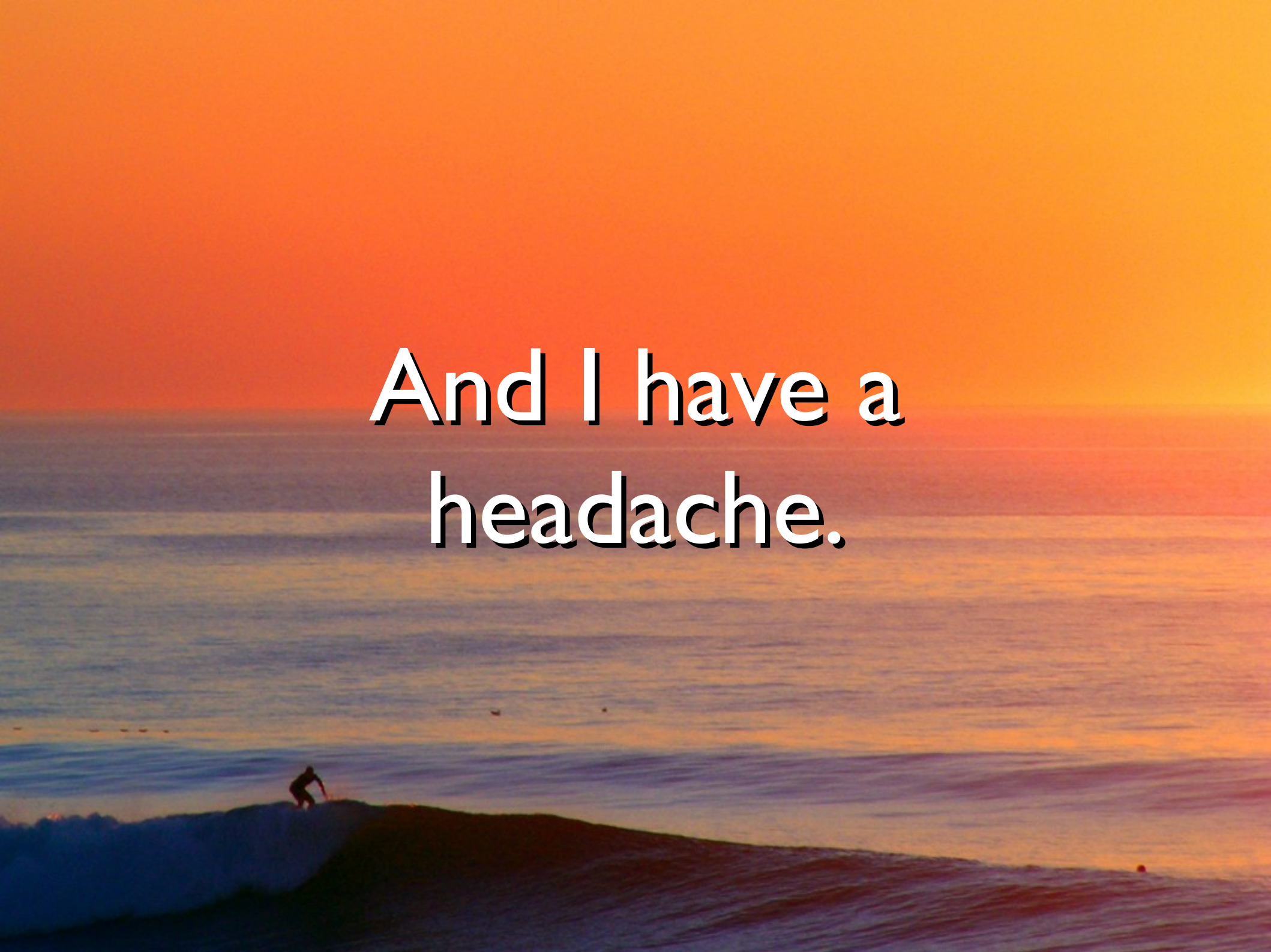
Real World Haskell.

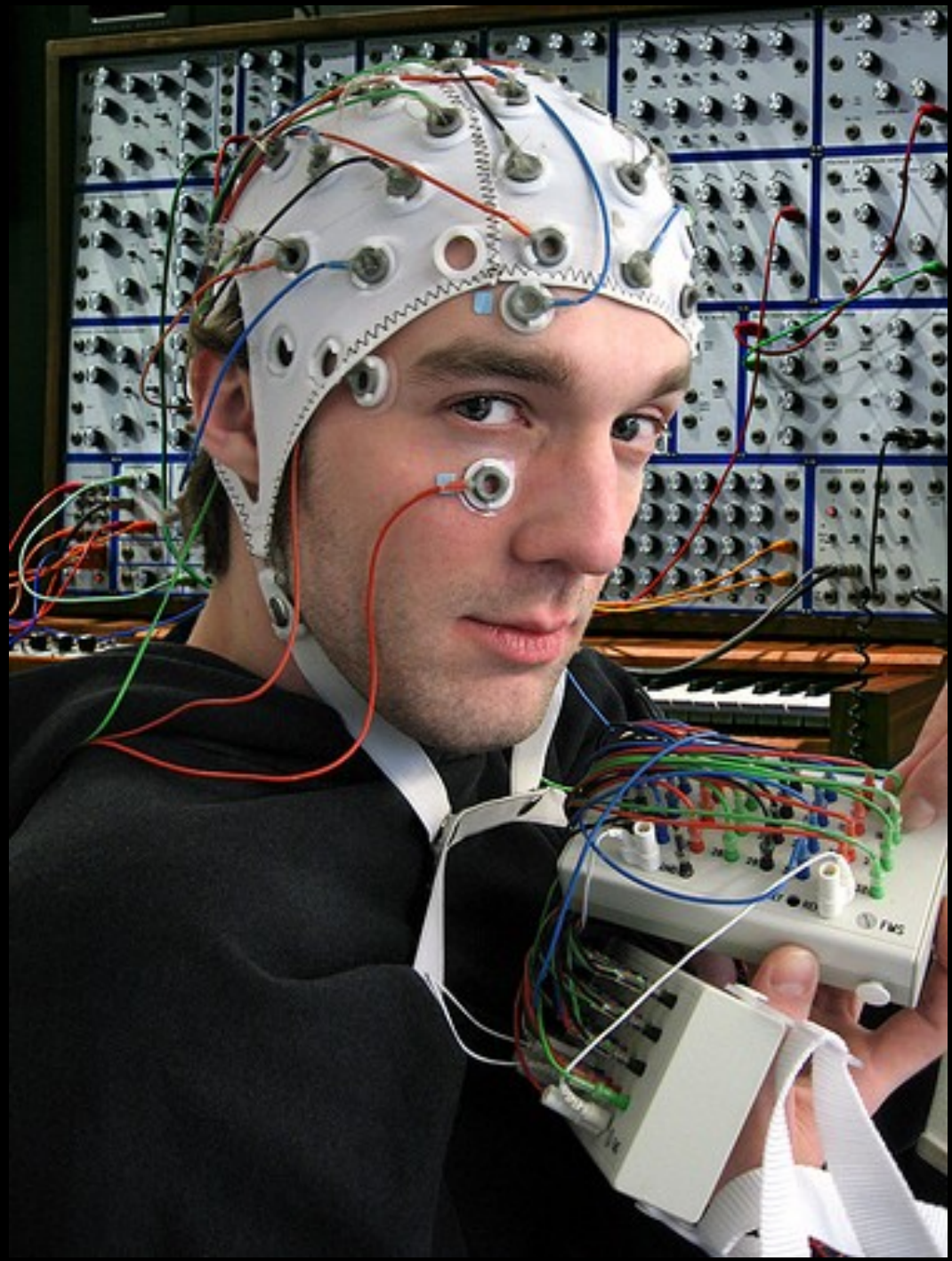
It's 1994.

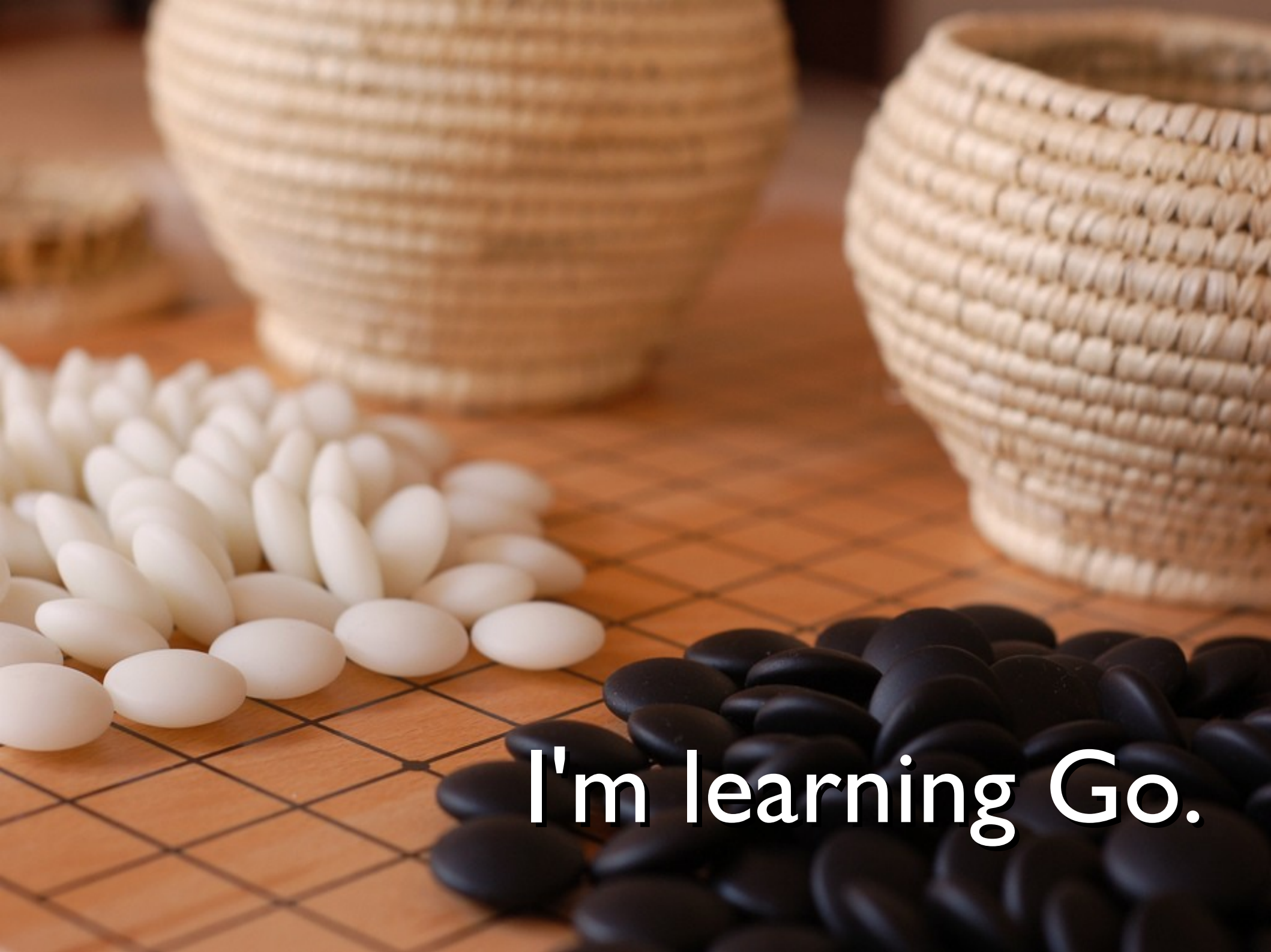
There's just one
song on the radio.
“Love is all around,”
by Wet Wet Wet.


And I have a
headache.

I'm in
Glasgow.

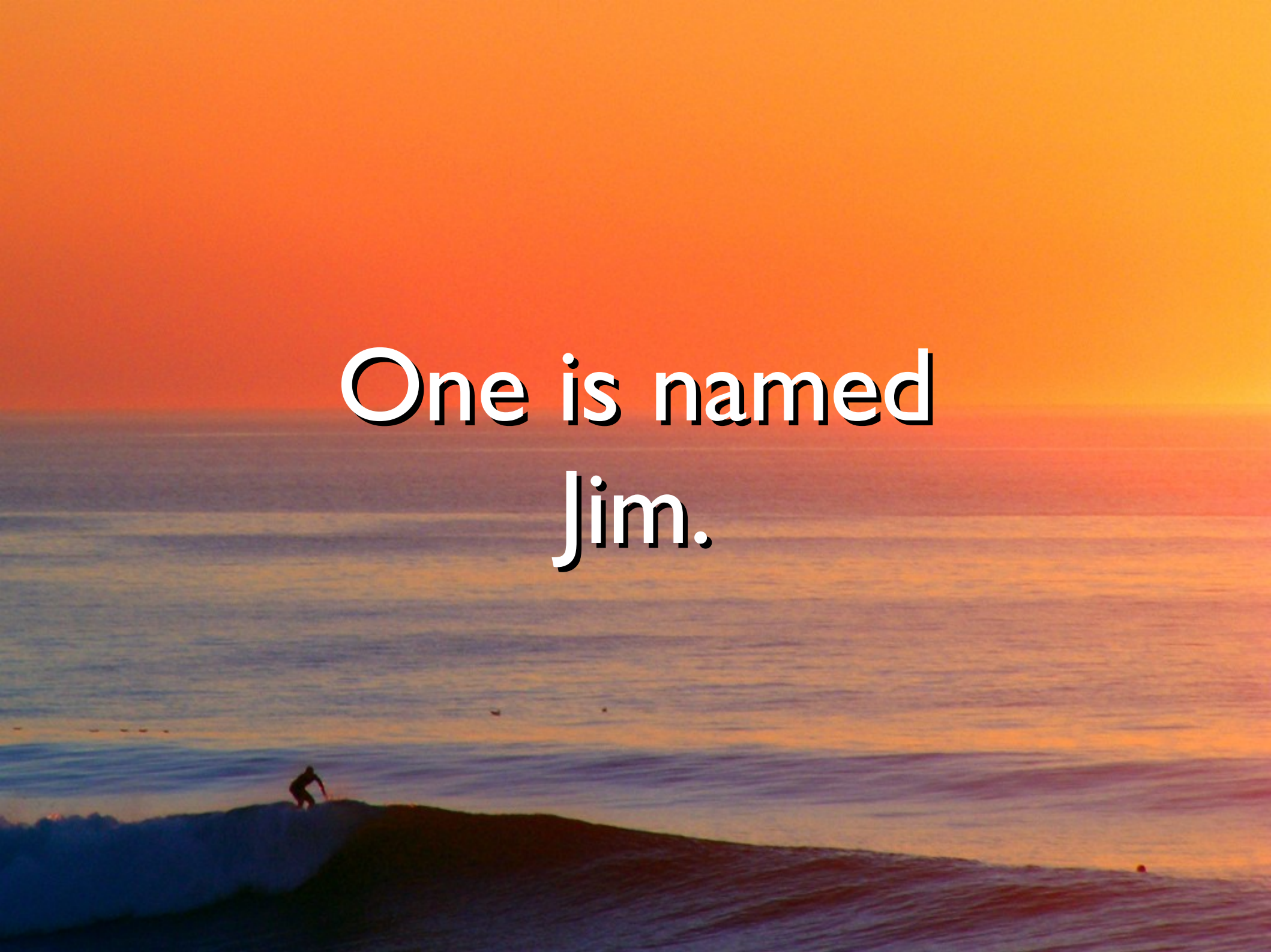
I'm learning to hack on GHC.

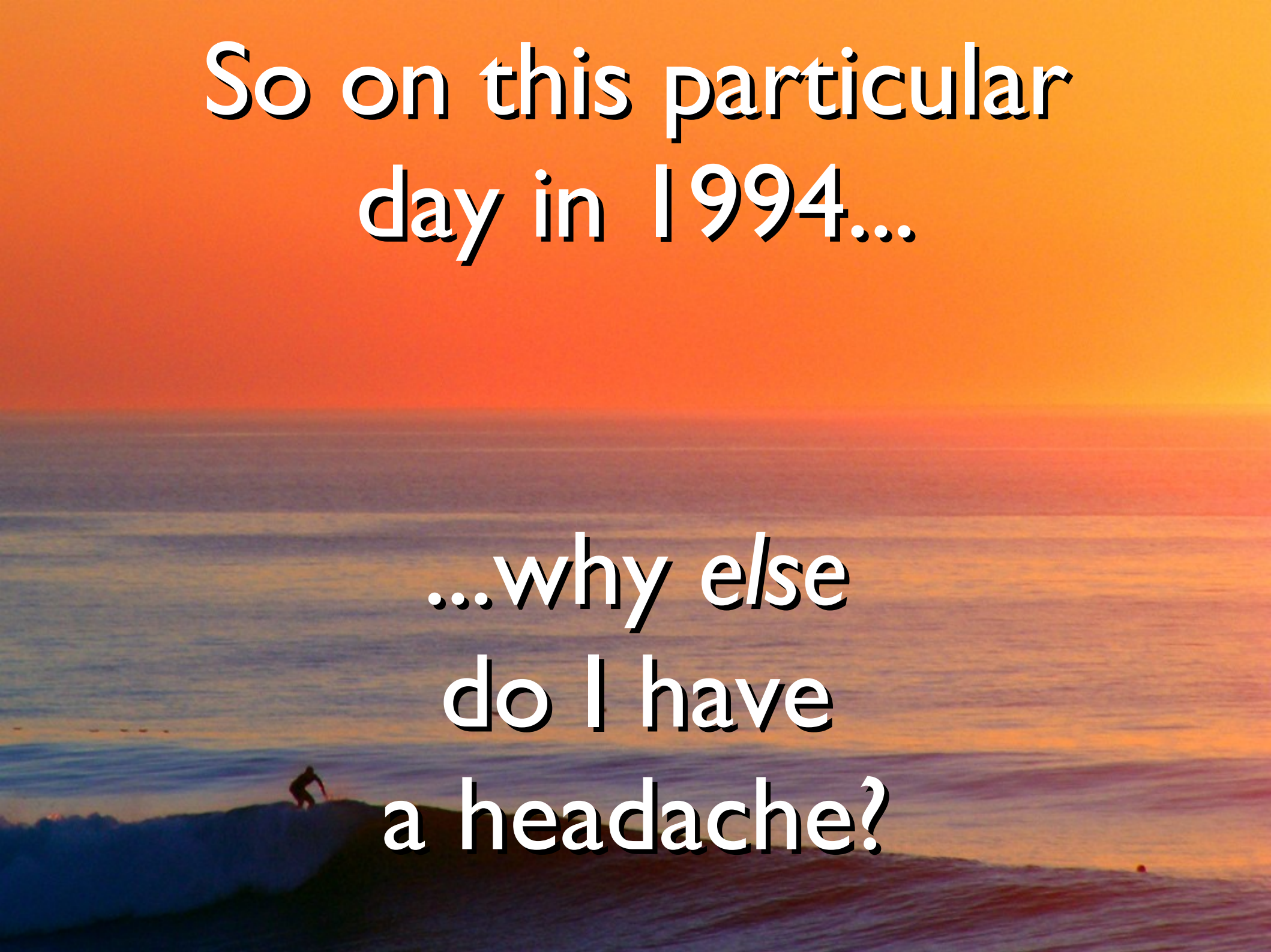



I'm learning Go.

It's a great crowd.

I make lots of friends.


One is named
Jim.


So on this particular
day in 1994...
...why *else*
do I have
a headache?

Well, we tend to go out.

After a hard night out, Jim goes home.

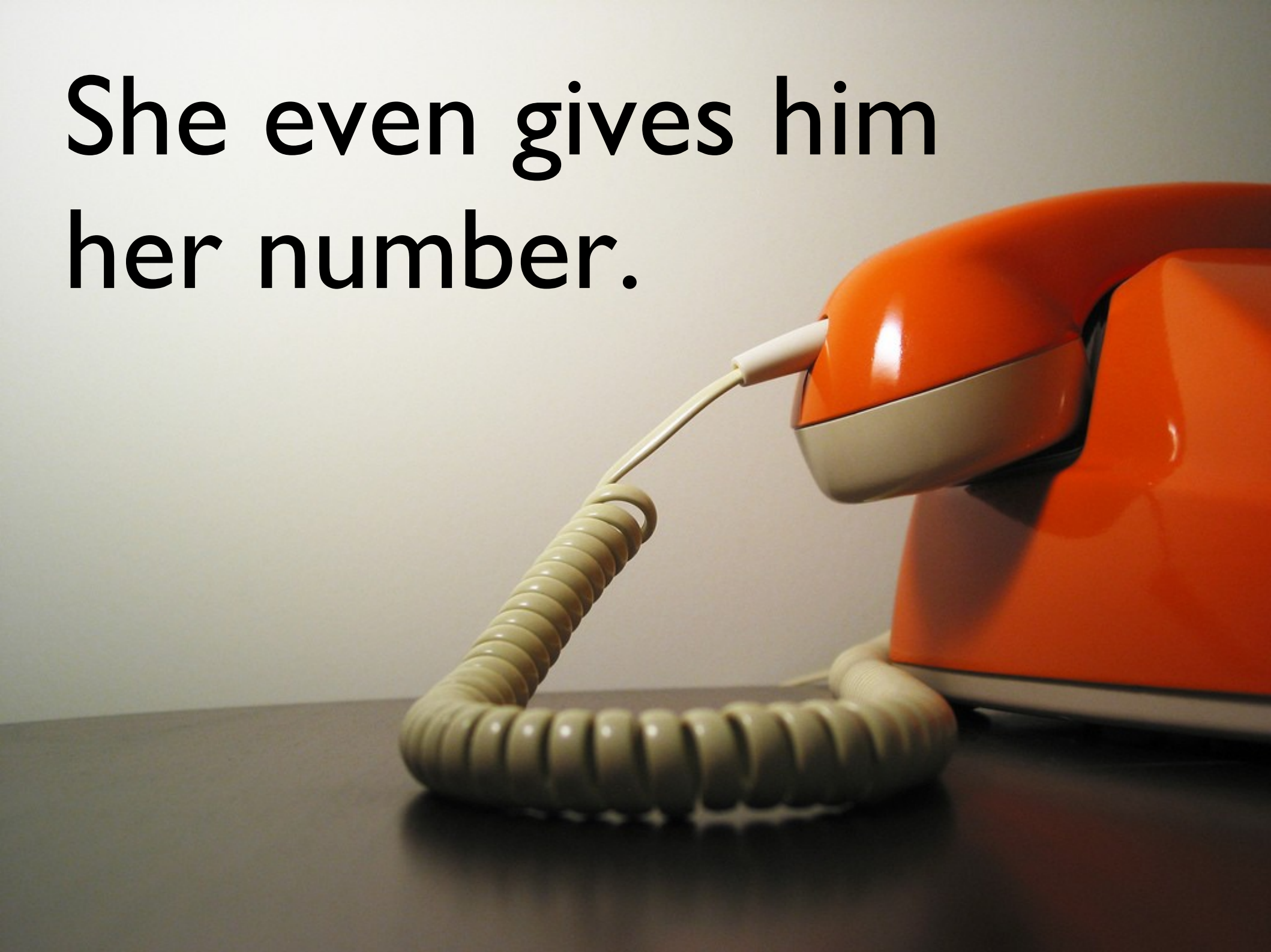
He dreams.

Happily, this is no nightmare.
In fact...

...he meets a
beautiful
woman.

In his dream,
they hit it off.

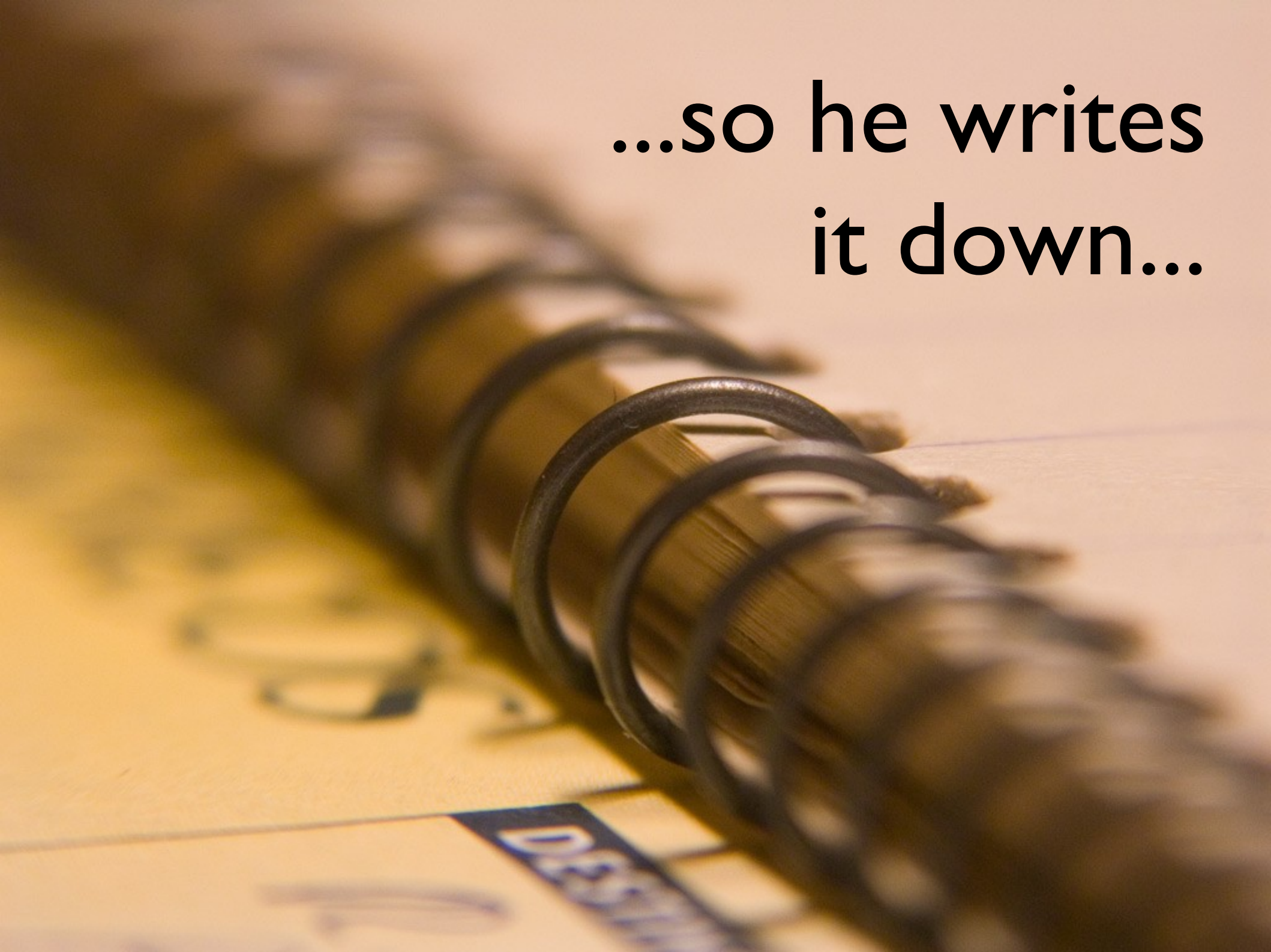
She even gives him her number.

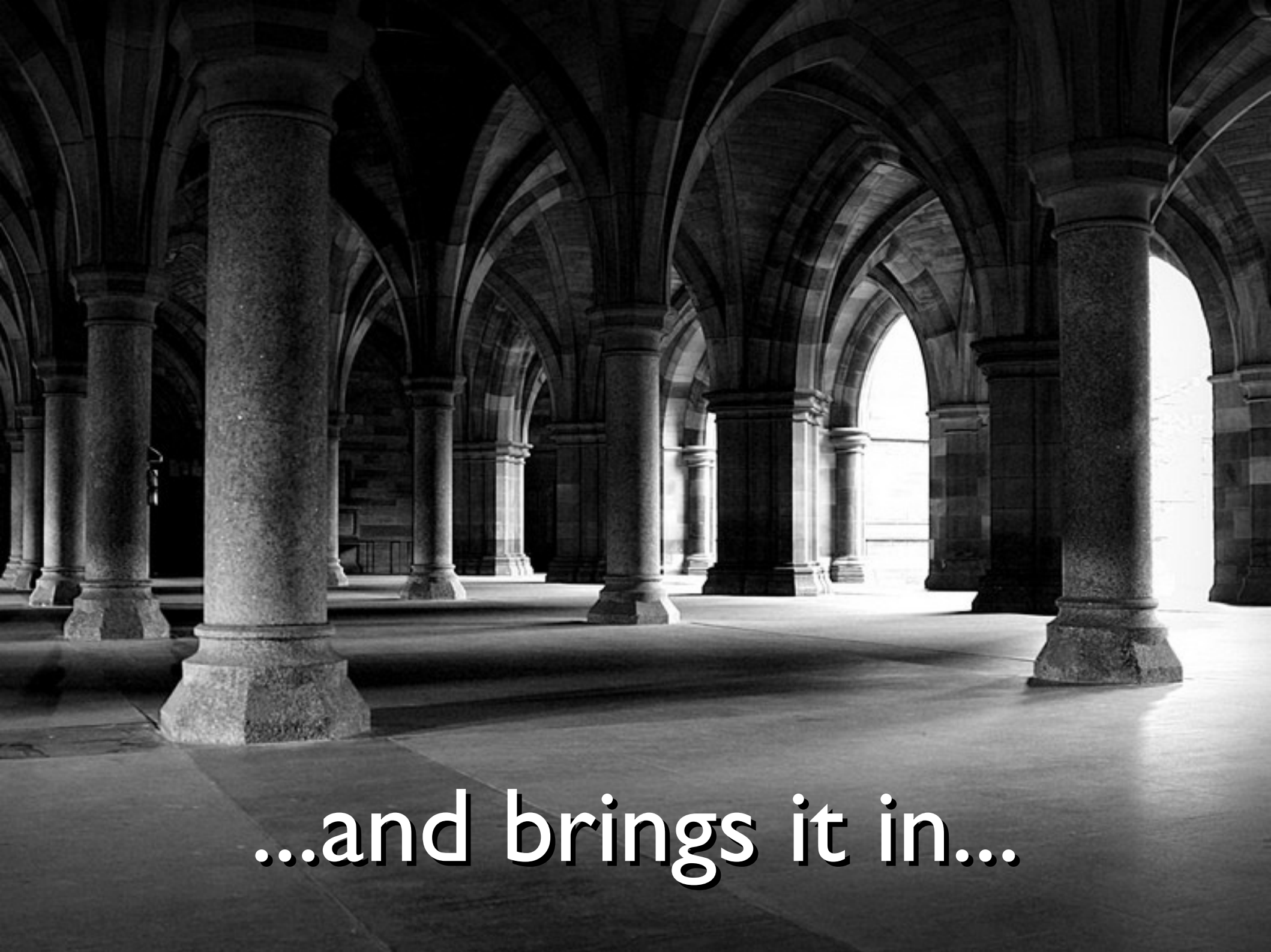


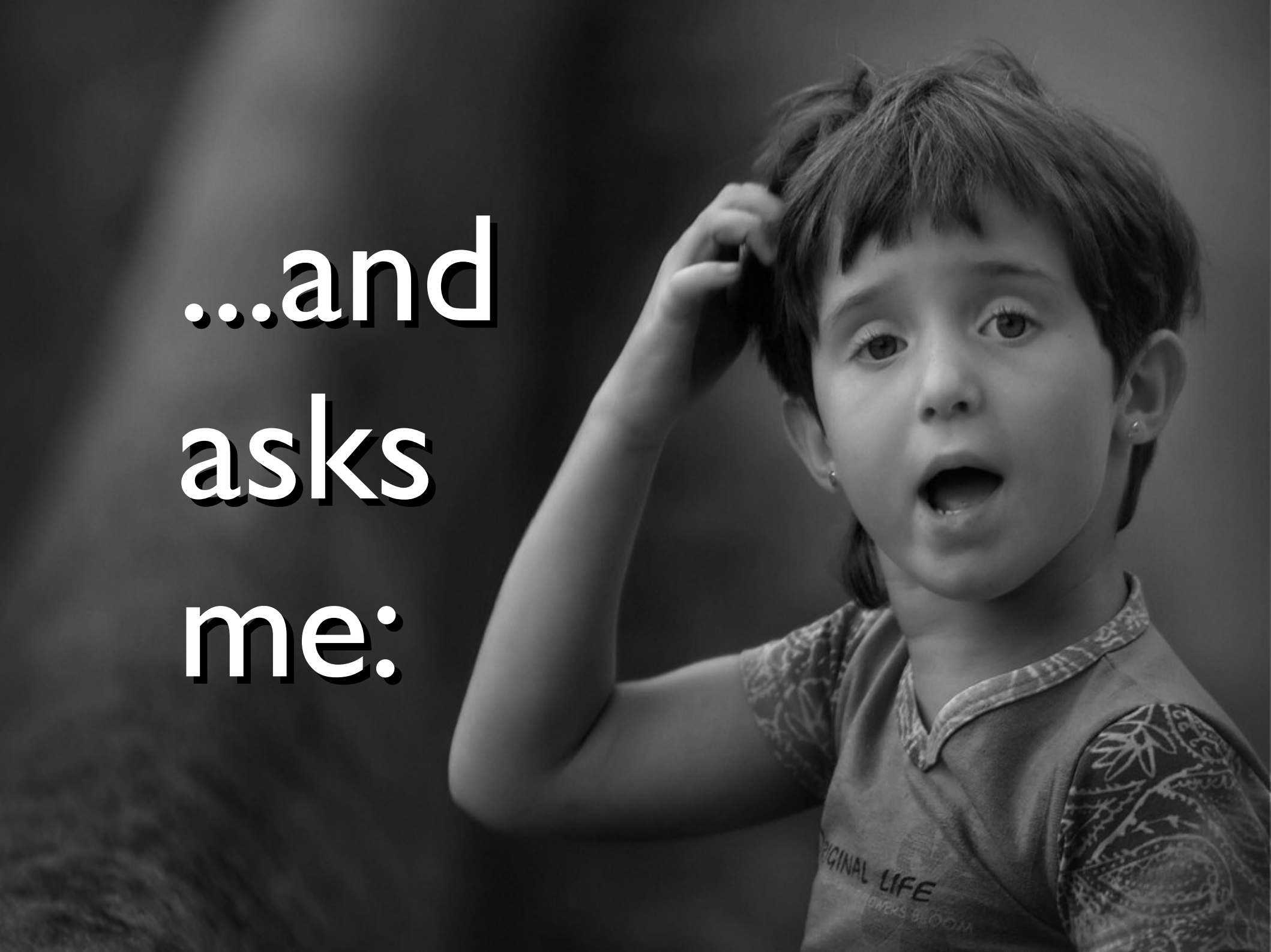
When he
wakes...

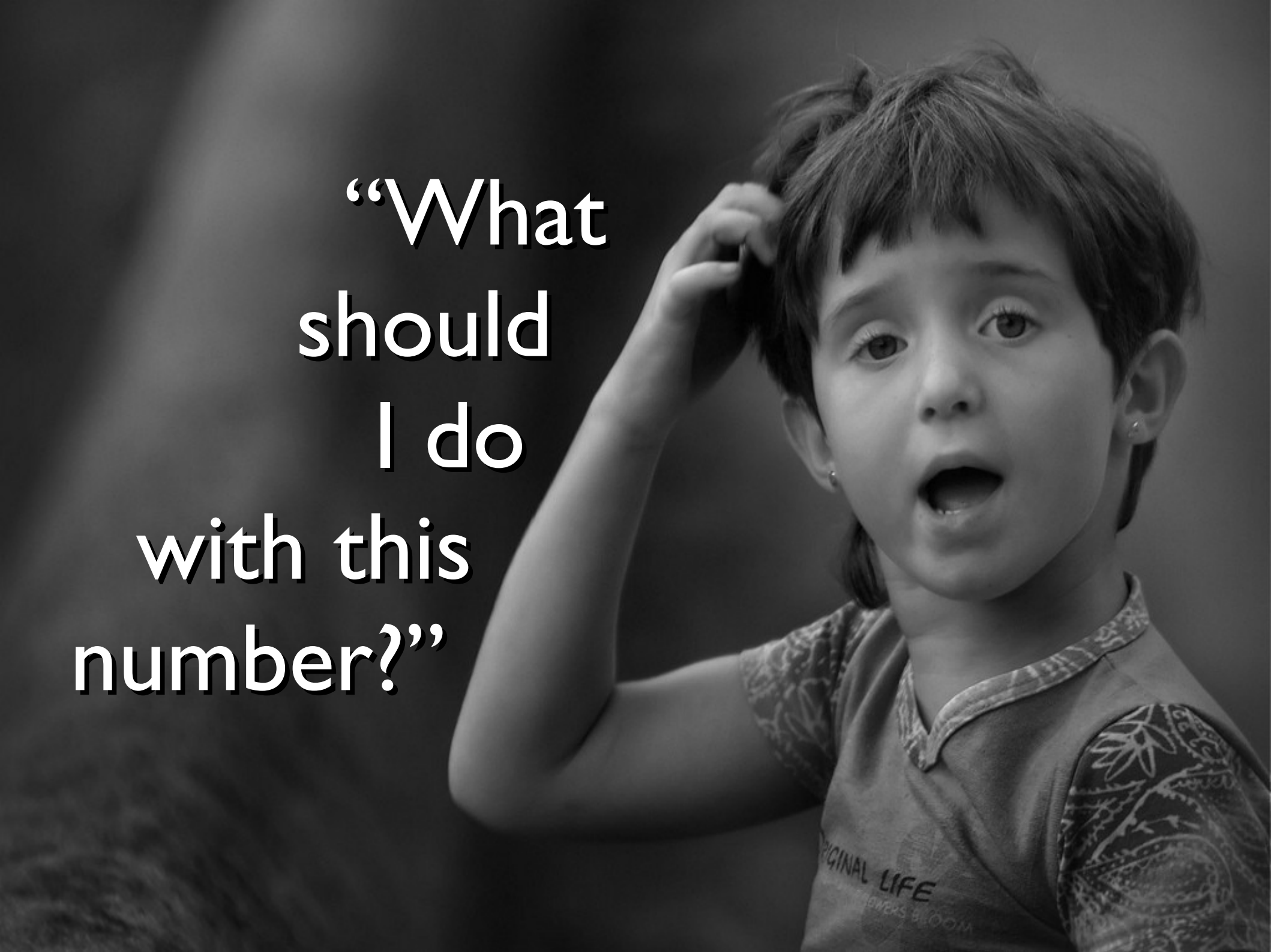
...it's the most.
Amazing.
Thing.

He *remembers*
her number!


...so he writes
it down...


...and brings it in...


...and
asks
me:


"What should I do with this number?"

“Wait a sec,”
I tell him.

“I *know* this number.”

“It's the delivery number...

...for the local pizzeria.”

Why tell *this* story?

Community.

Most people
like to
belong.

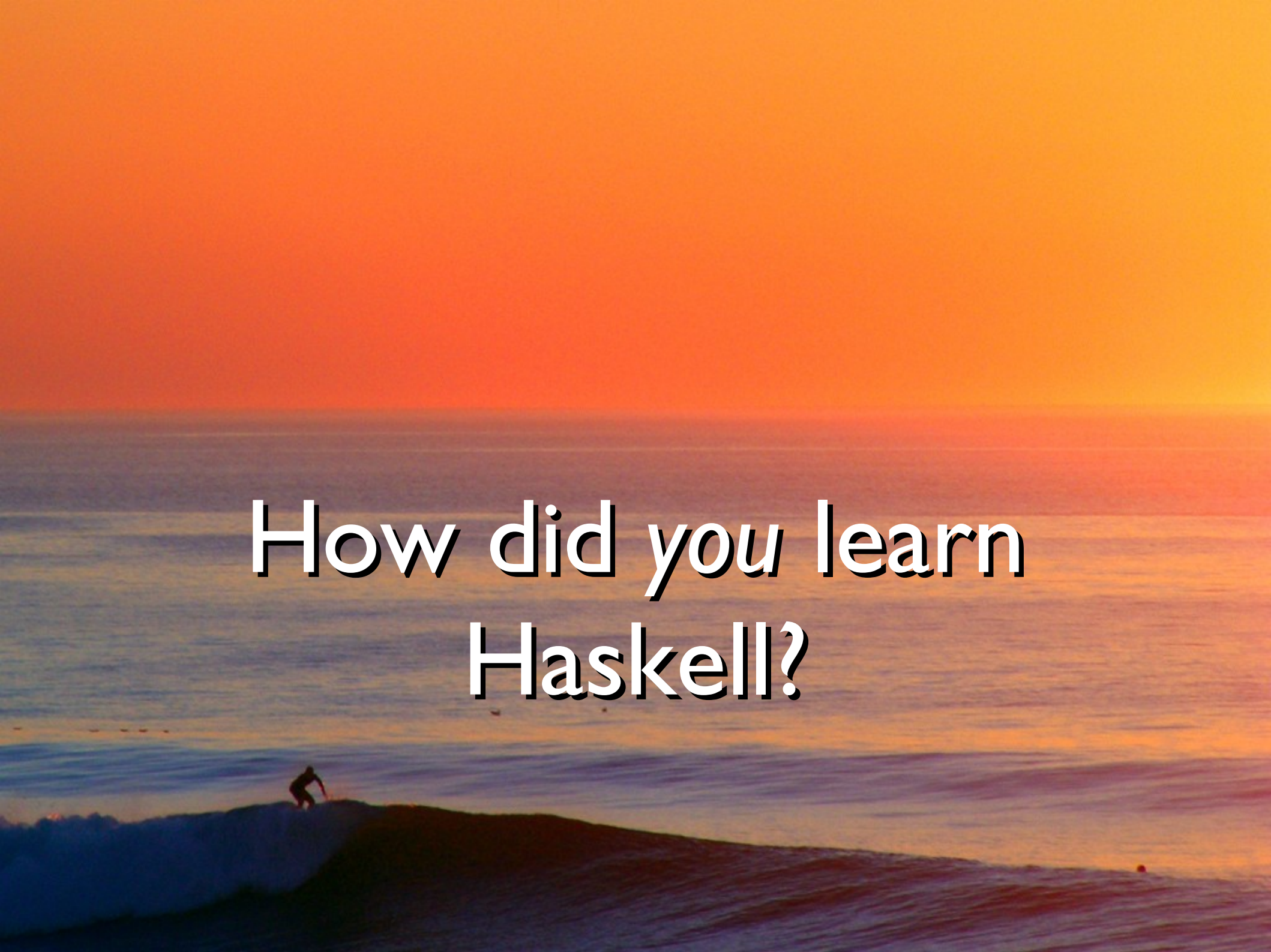
I loved
the Haskell
community...

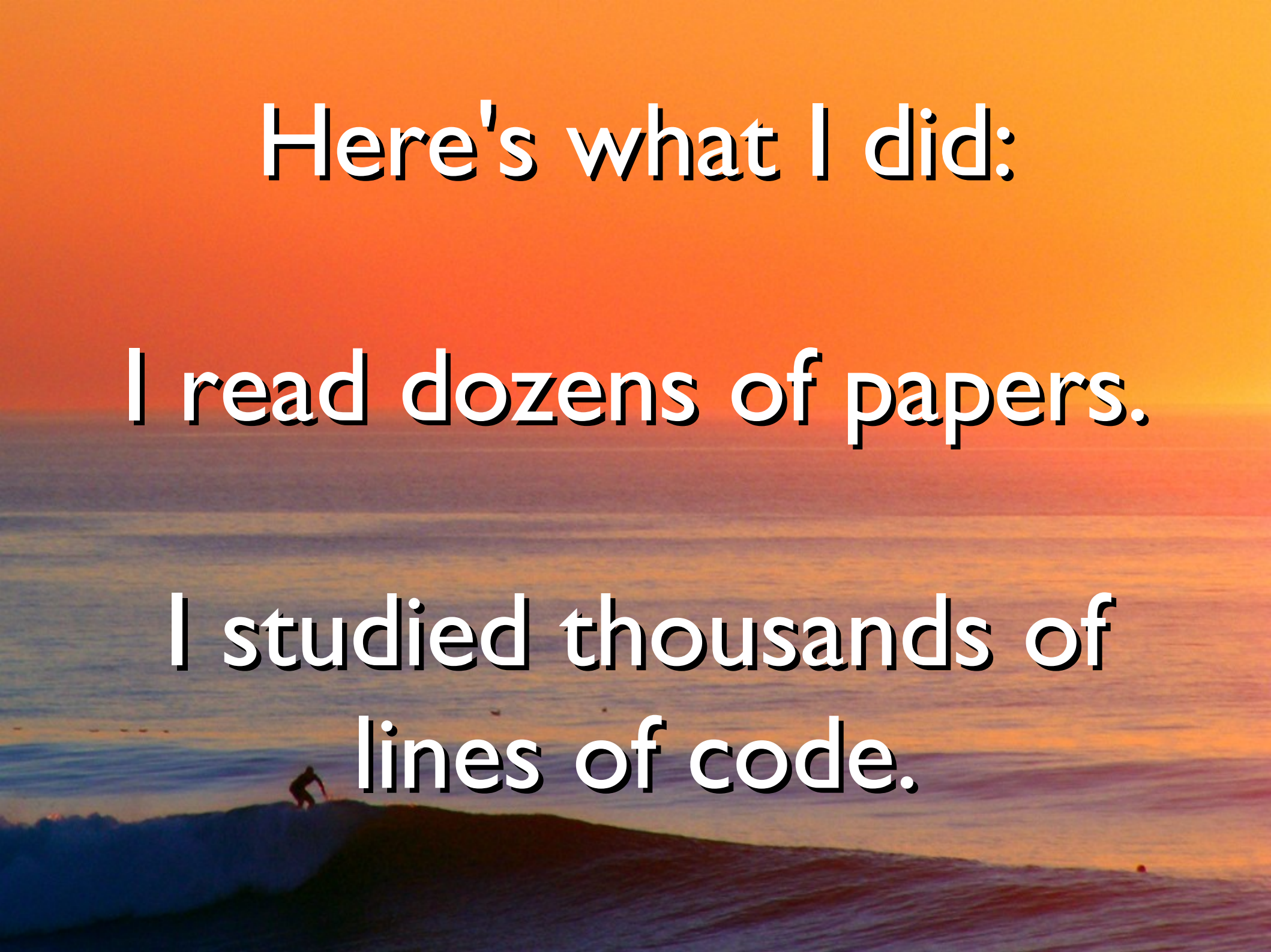
...but
I left
anyway.

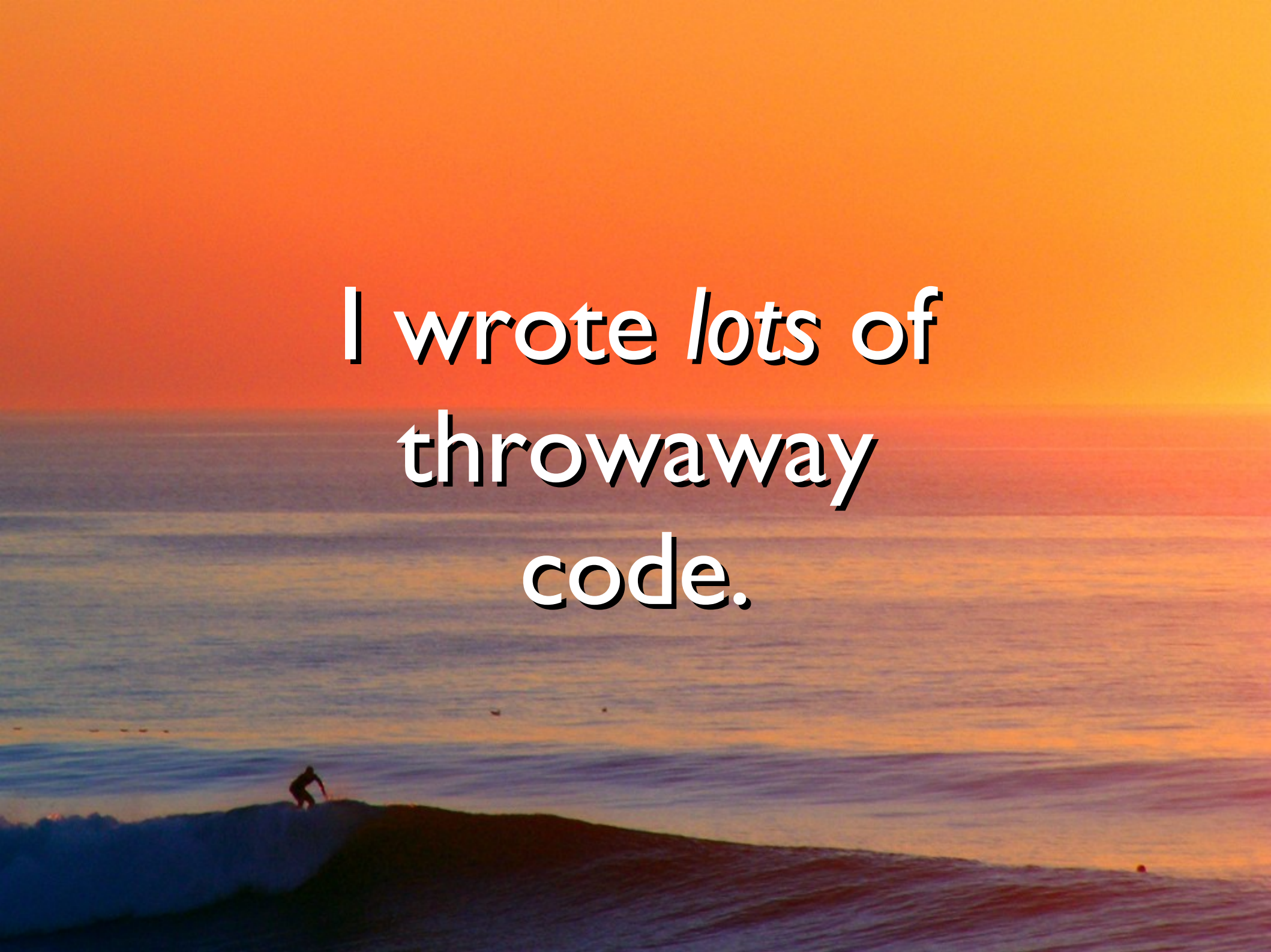
Over a decade
passed.

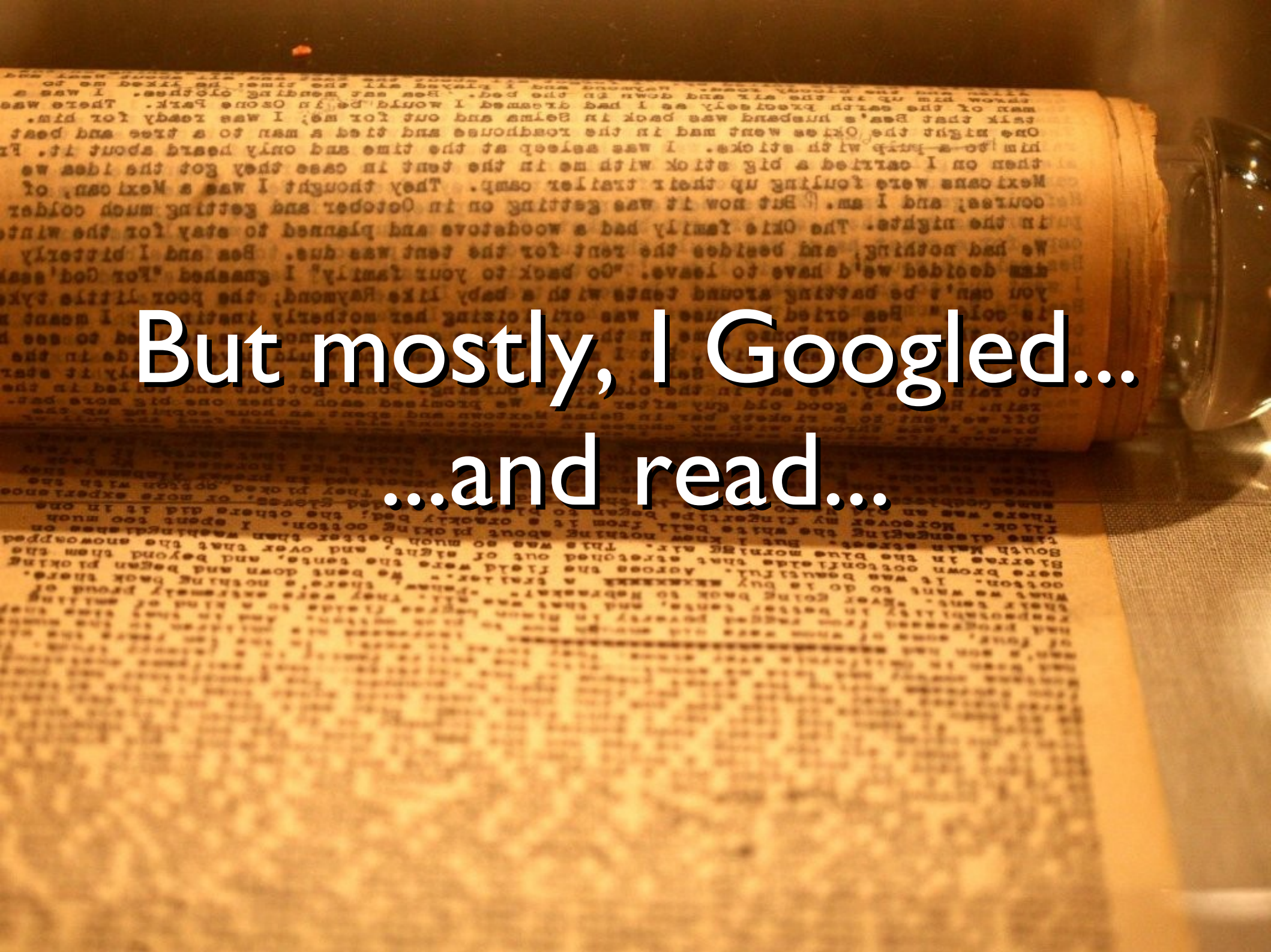
When I returned, the
community was still *beautiful*...

...and serenely inward-facing.

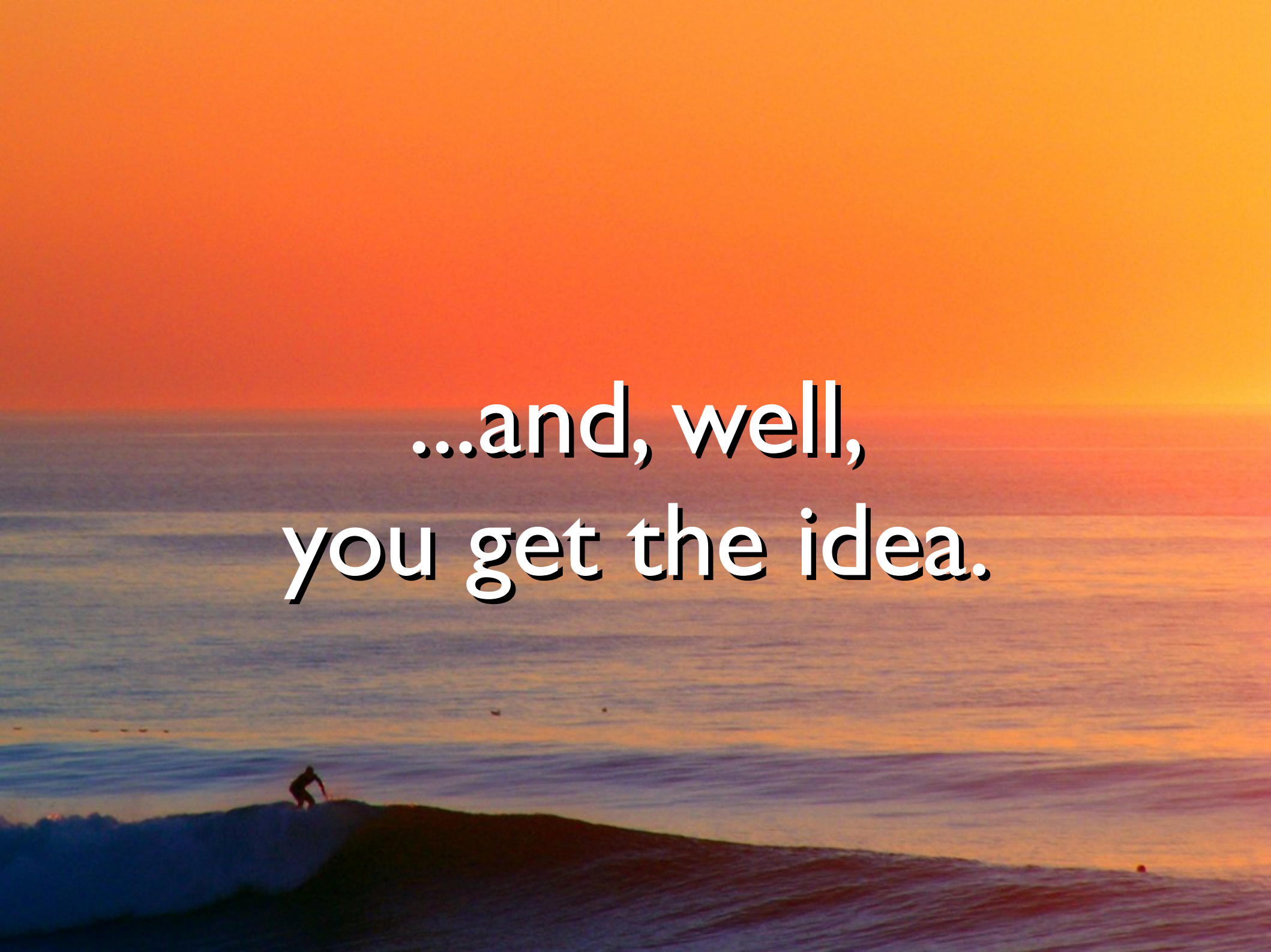

How did *you* learn Haskell?


Here's what I did:
I read dozens of papers.
I studied thousands of lines of code.


I wrote *lots* of
throwaway
code.

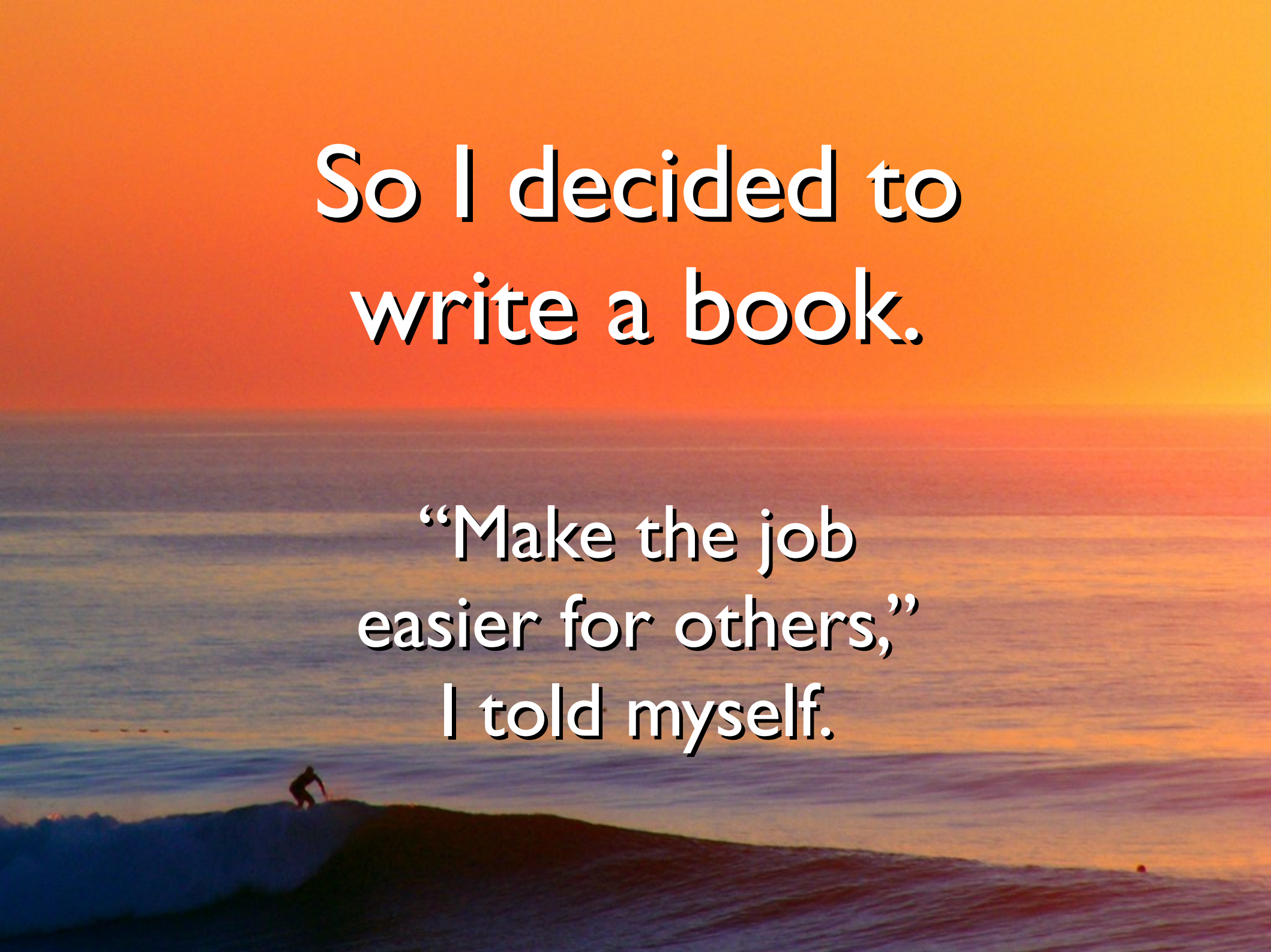

But mostly, I Googled...
...and read...

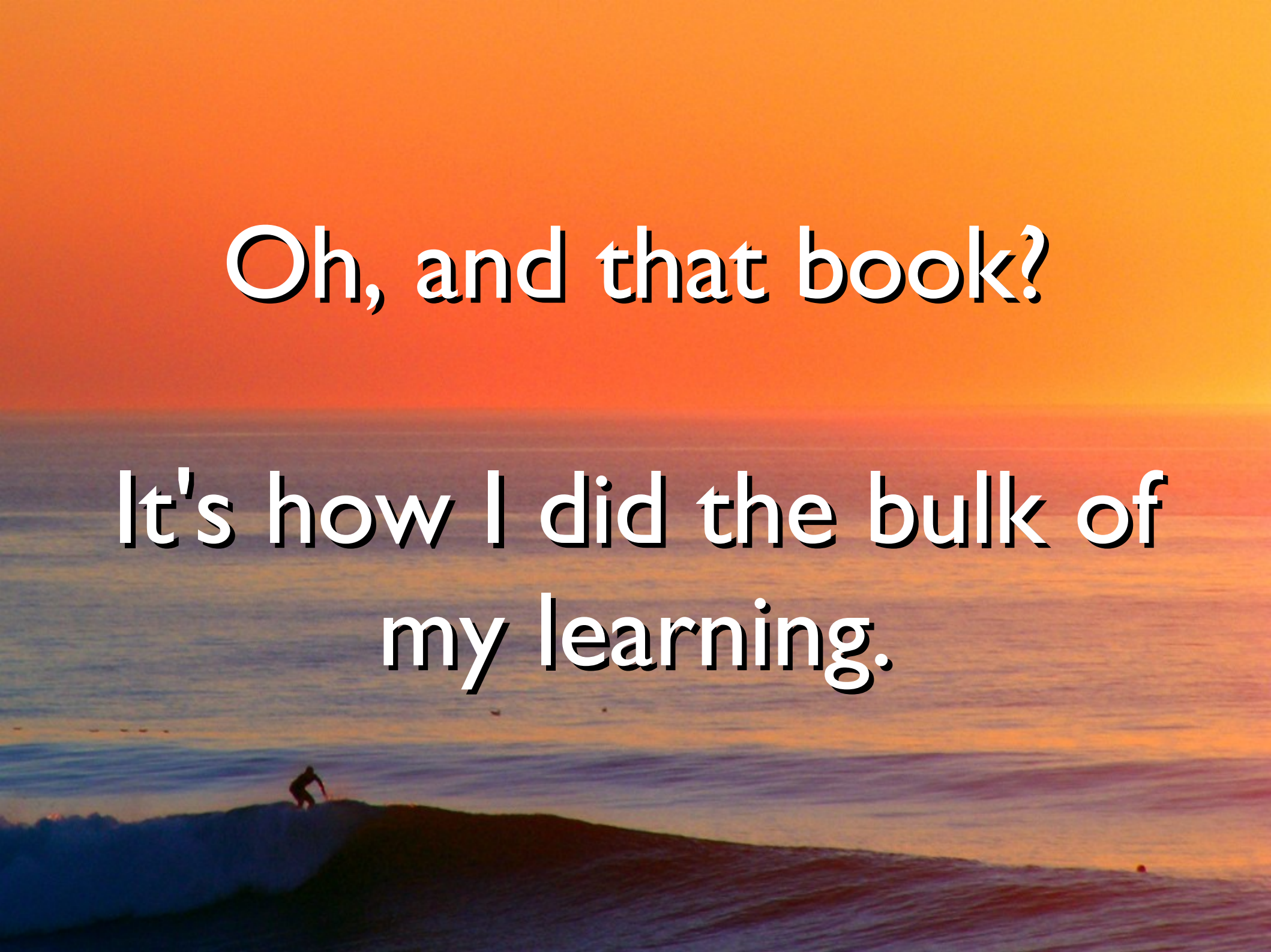
...and Googled...
...and read...
666
GAL. DE
eiusdem sunt generis,
guerint musculis: quid
que sinum magnum
tüs rariora fuissent?
gumentum, quòd
tibus ossa sunt laxura


...and, well,
you get the idea.

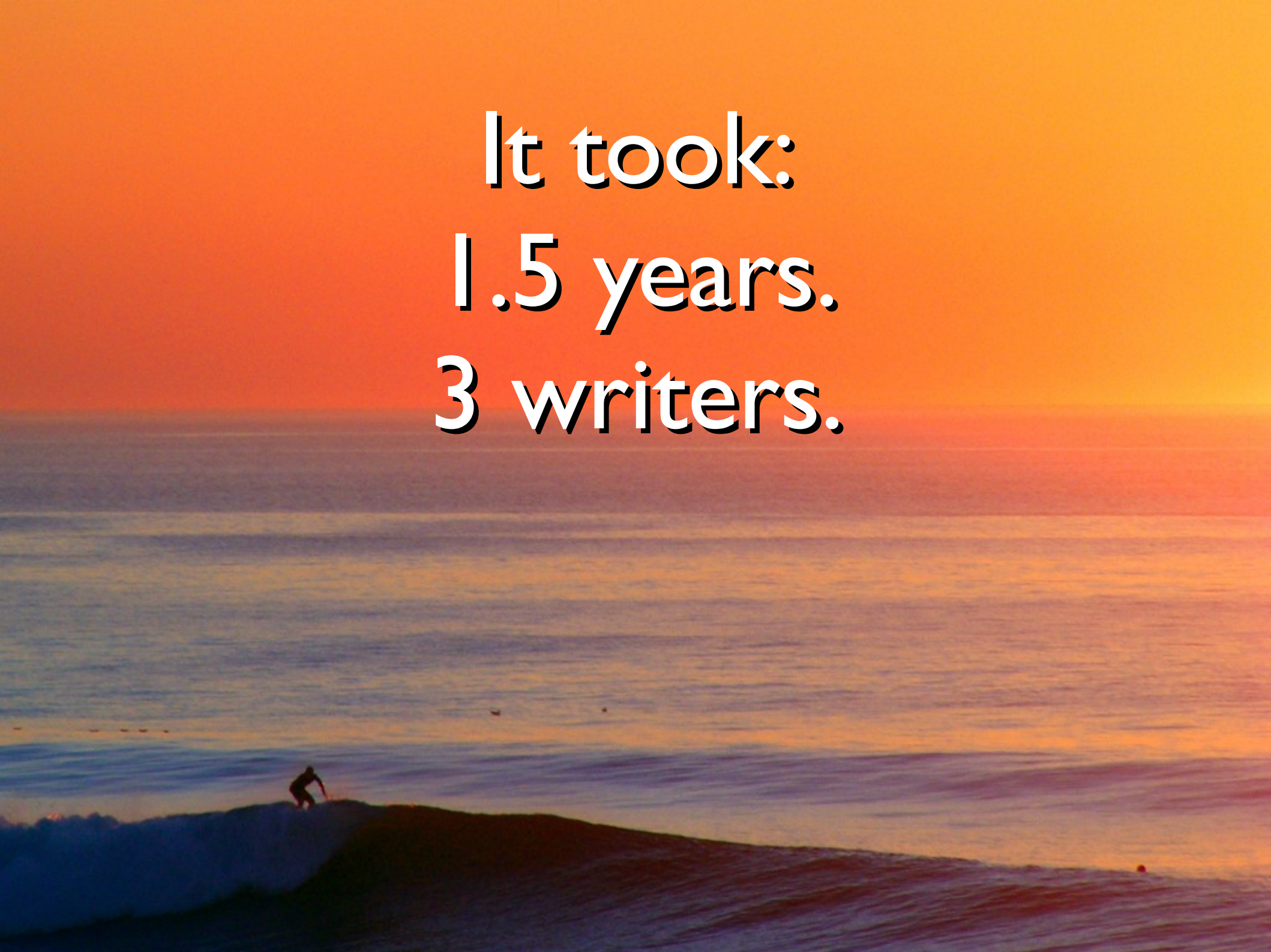
I delighted in the process.

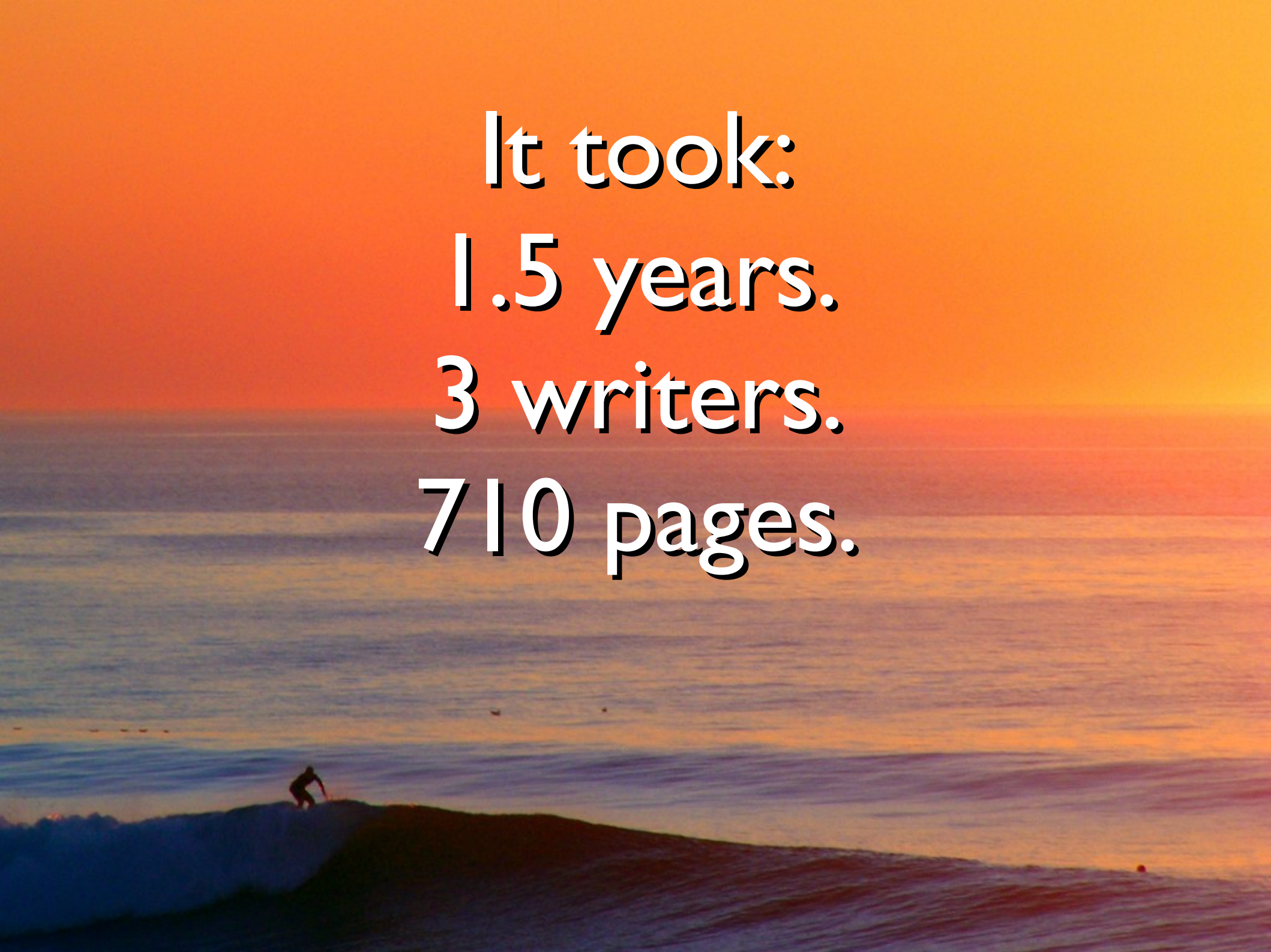
But it wasn't very efficient.

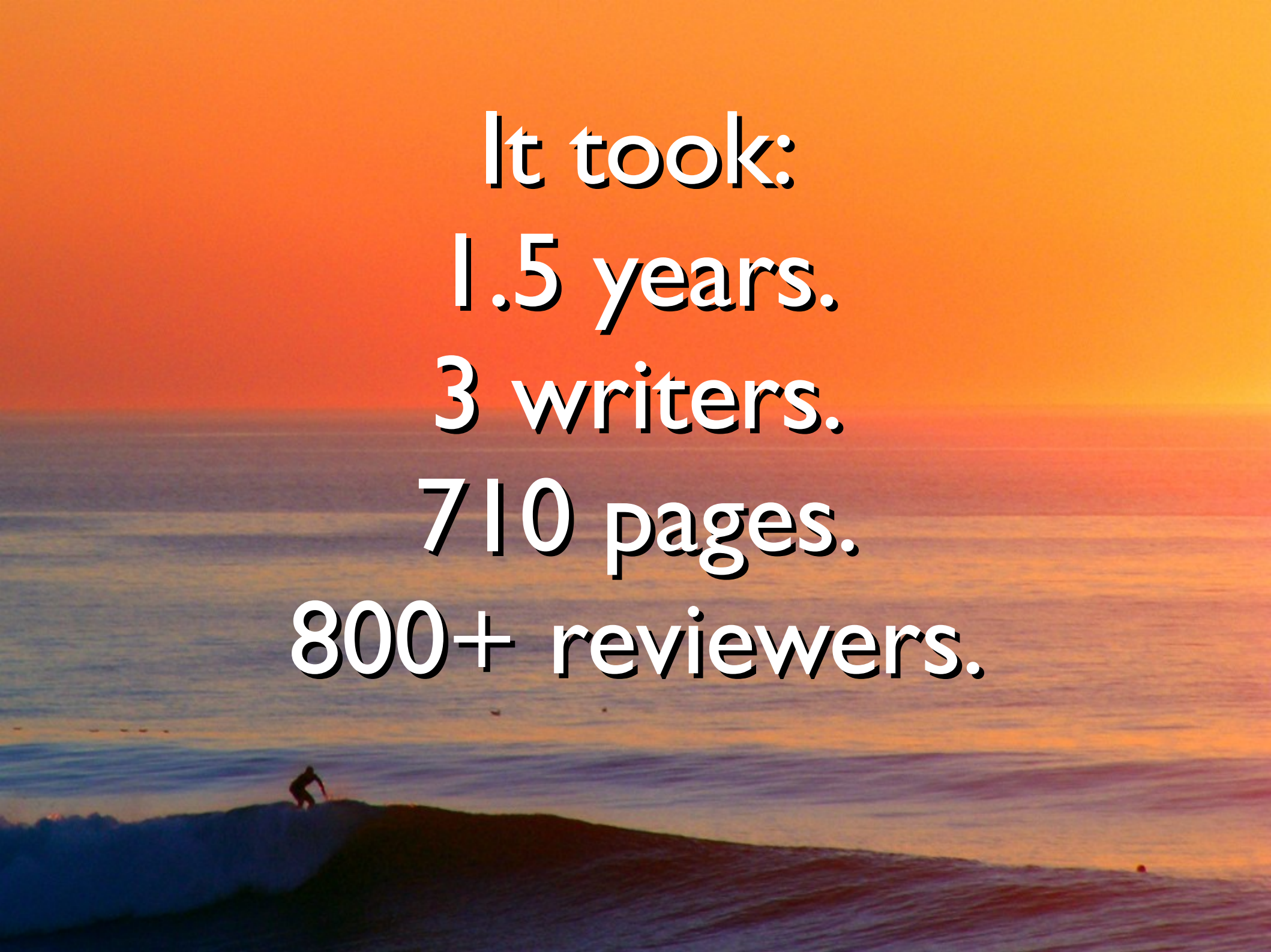

So I decided to
write a book.
“Make the job
easier for others,”
I told myself.


Oh, and that book?
It's how I did the bulk of my learning.

It took:
1.5 years.


It took:
1.5 years.
3 writers.


It took:
1.5 years.
3 writers.
710 pages.


It took:
1.5 years.
3 writers.
710 pages.
800+ reviewers.

It took:
1.5 years.
3 writers.
710 pages.
800+ reviewers.
7,500+ comments.

Now it's your turn.

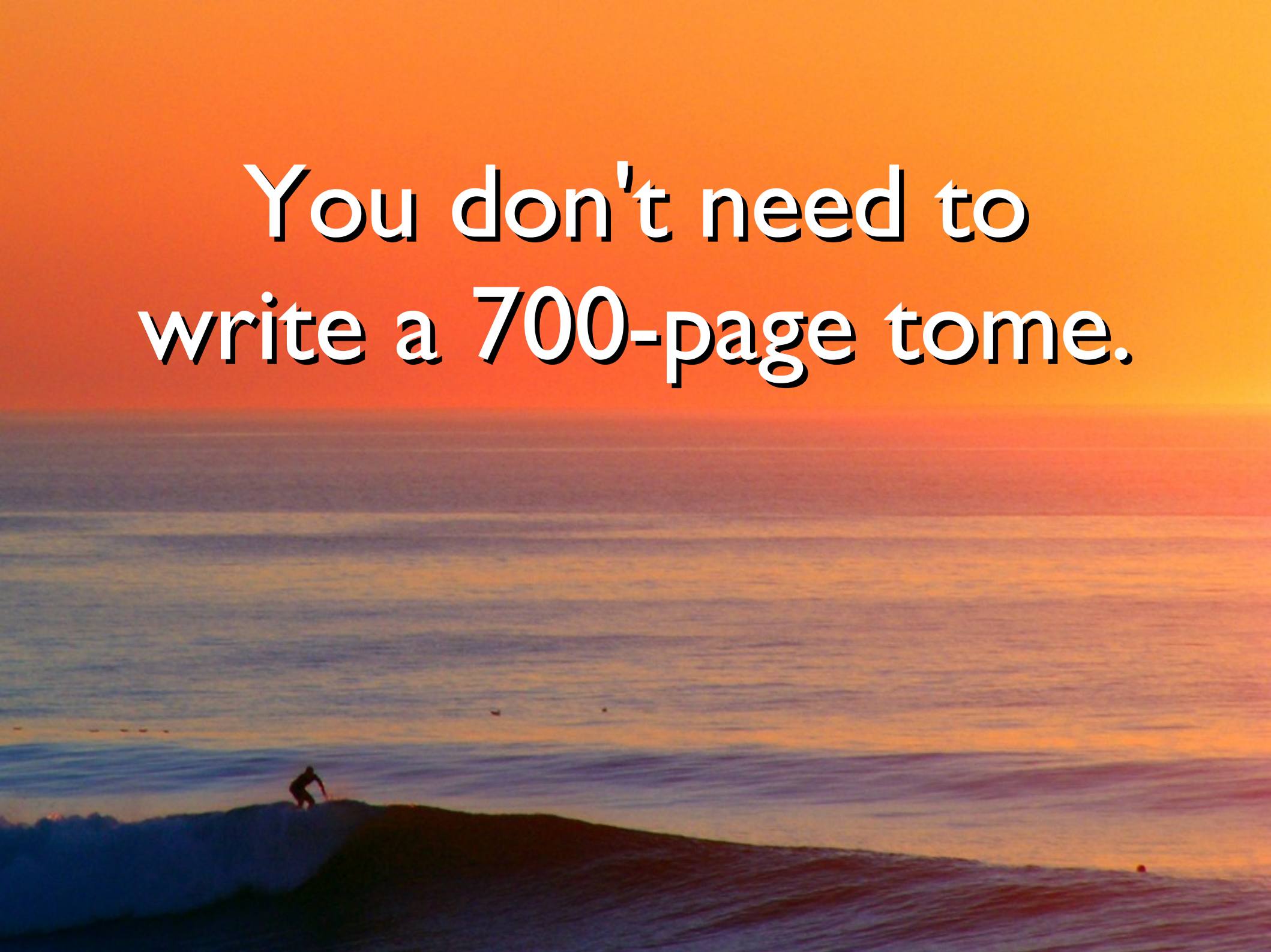
Srsly.

What is the Haskell community good at?

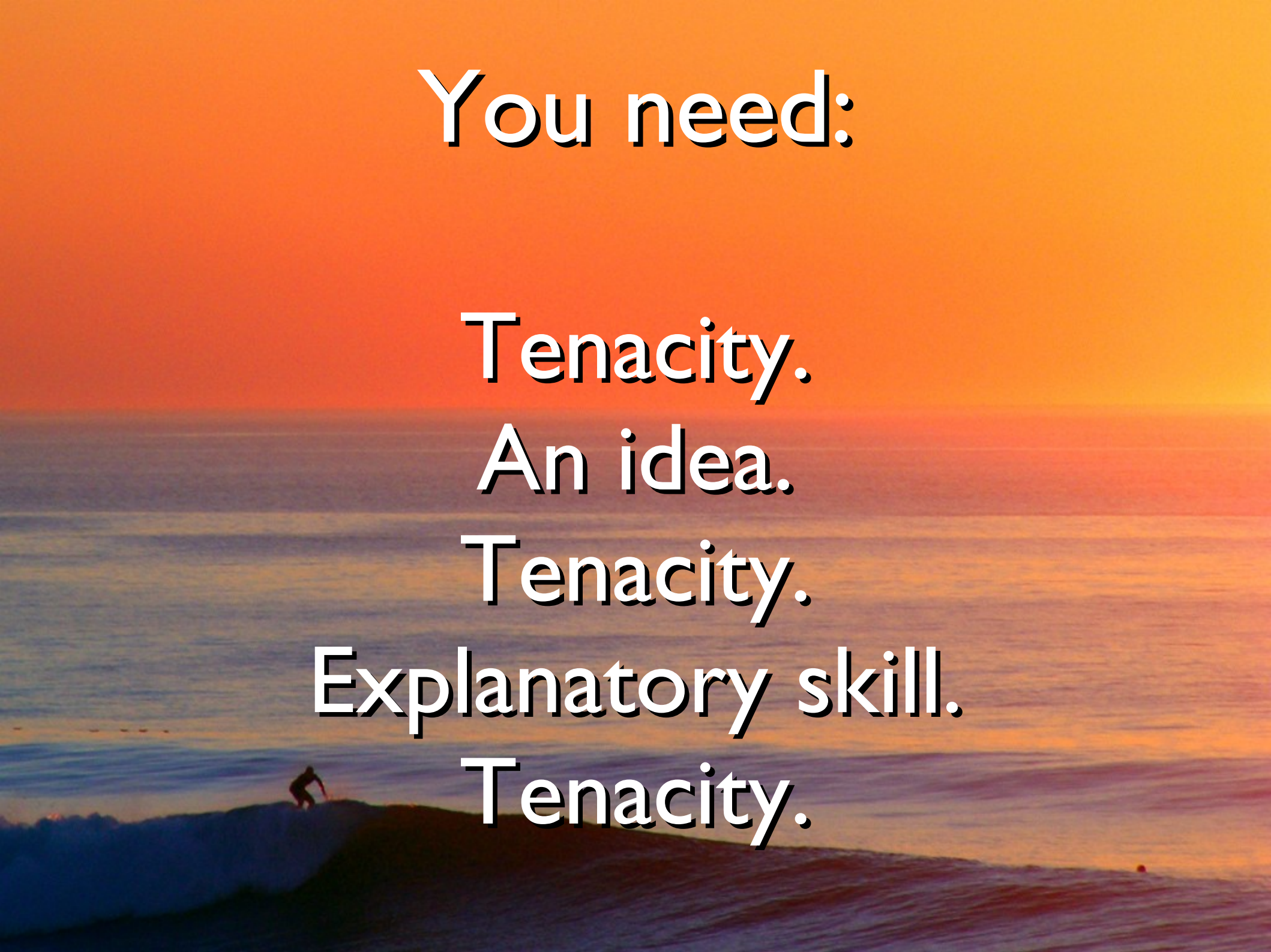
Research? We've got it!

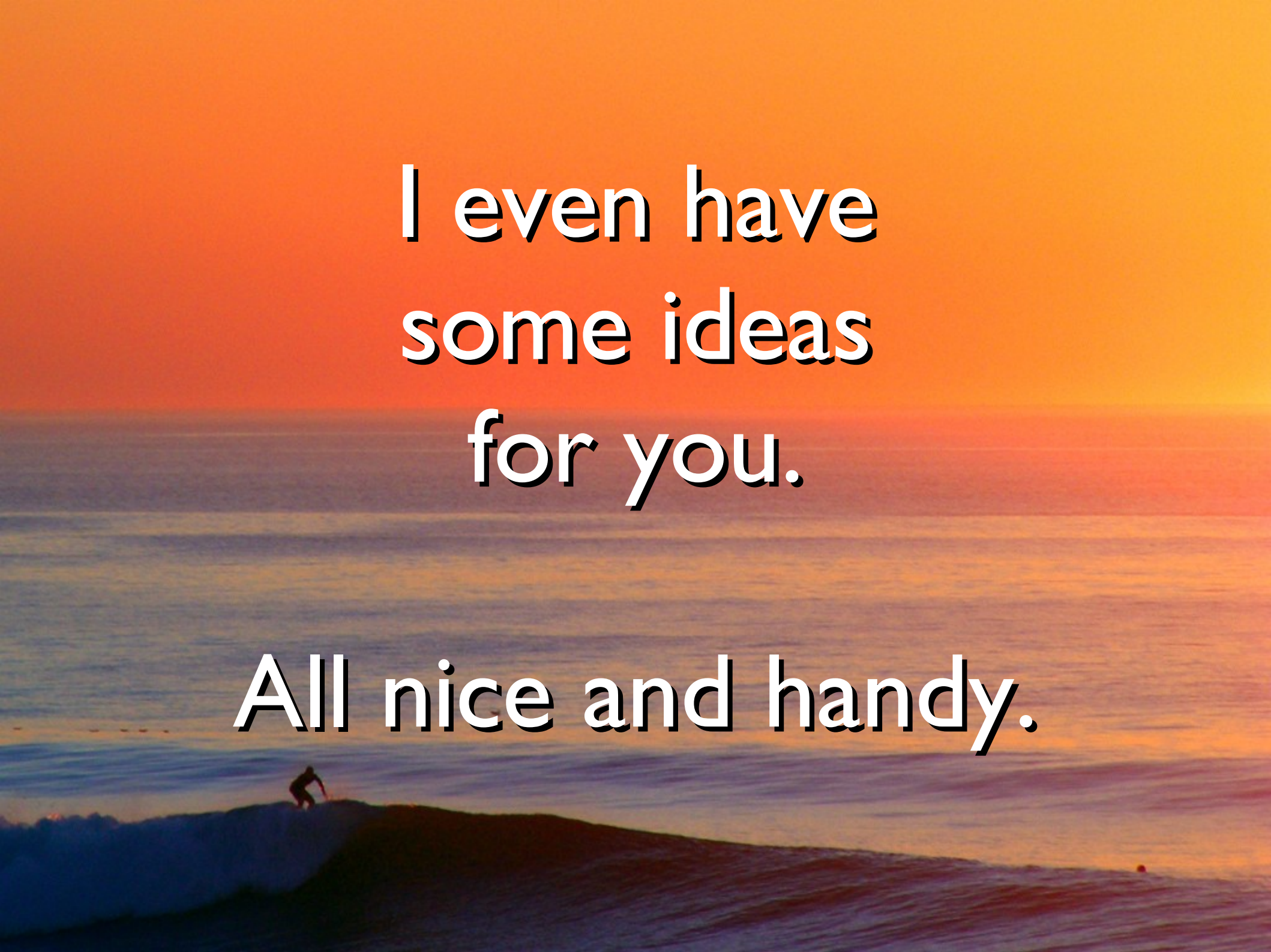
Outreach?

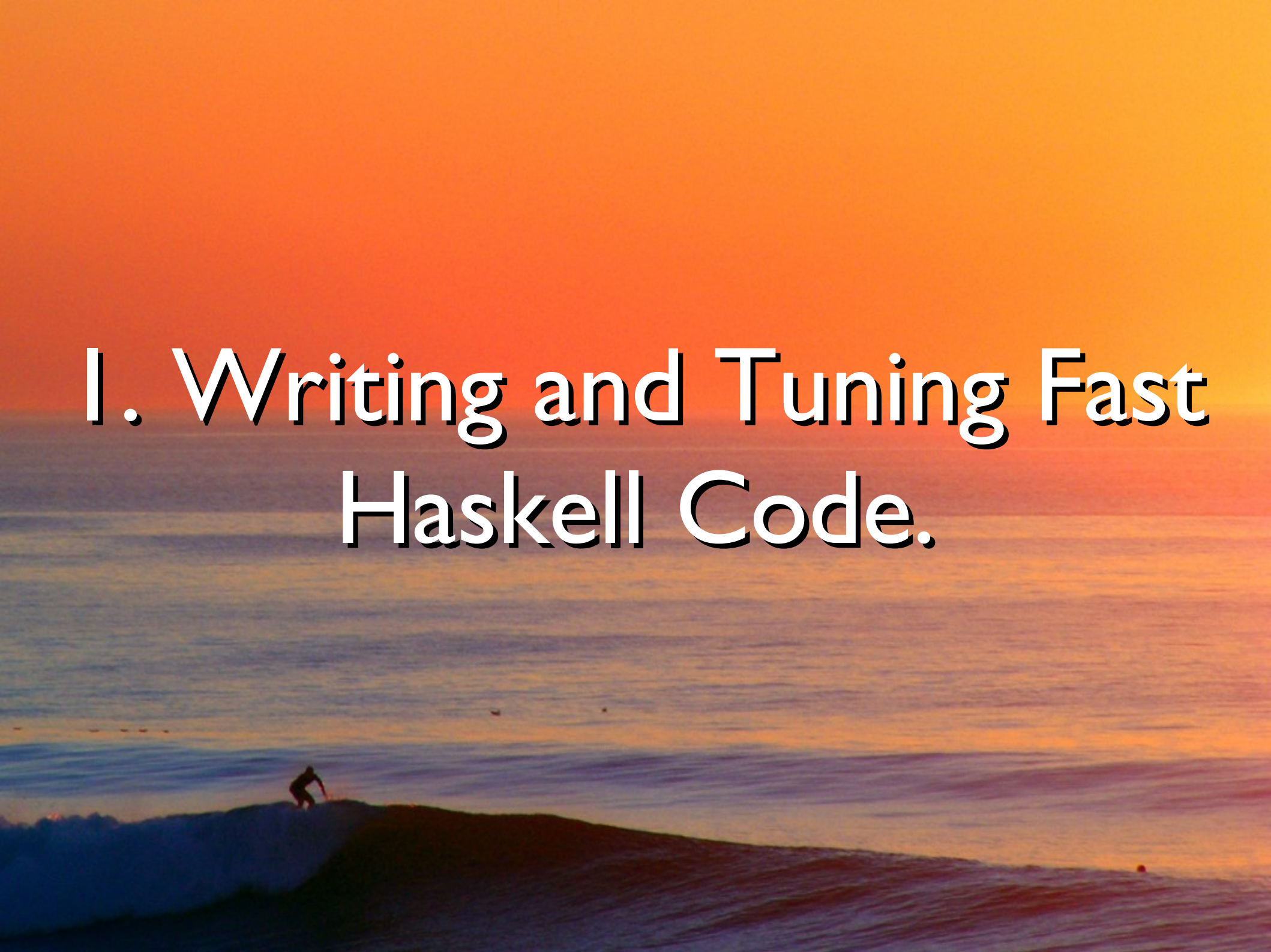
That's less clear to me.

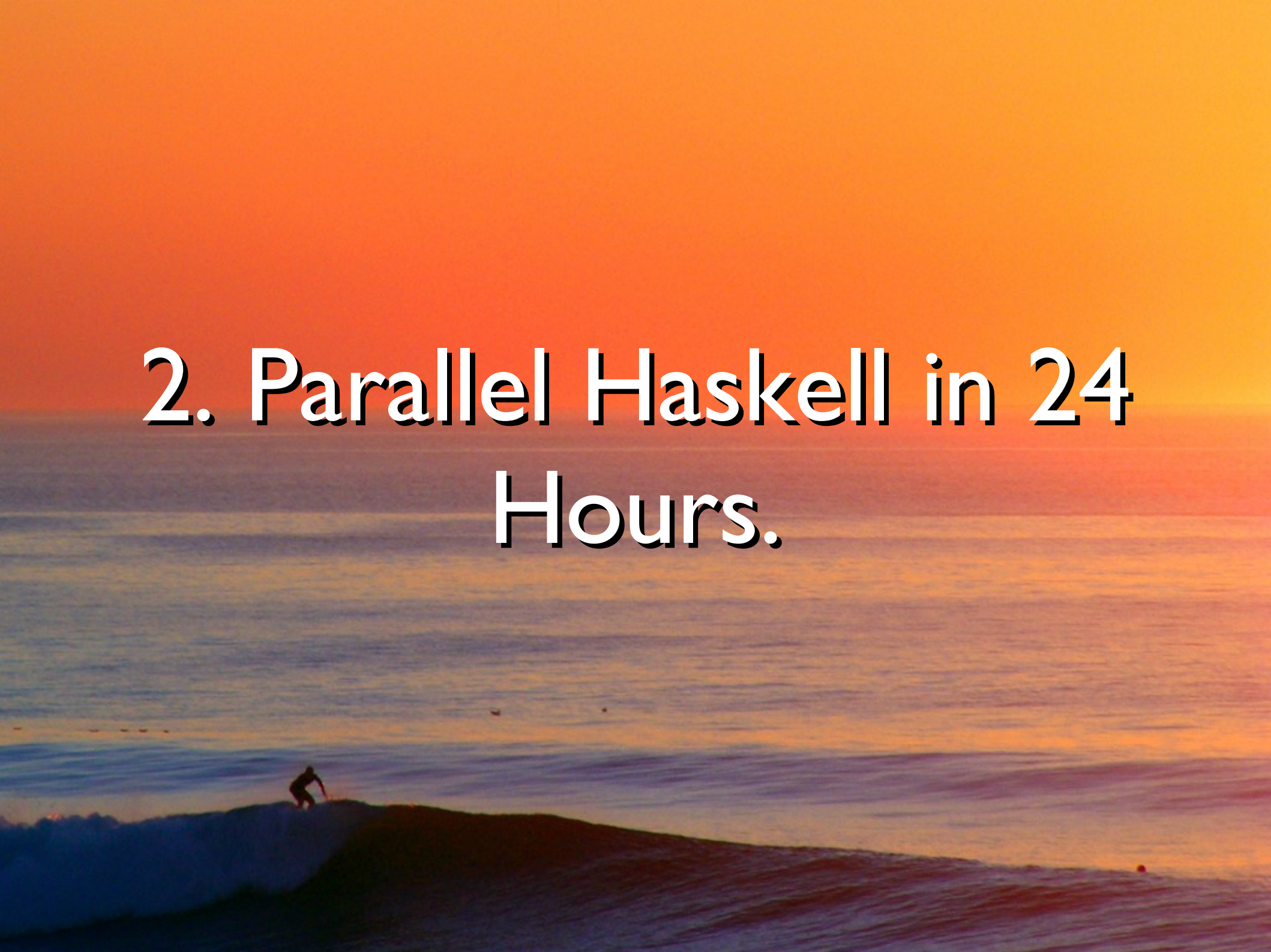

You don't need to
write a 700-page tome.

You don't need to describe new research.
(In fact, I think it's best you don't.)

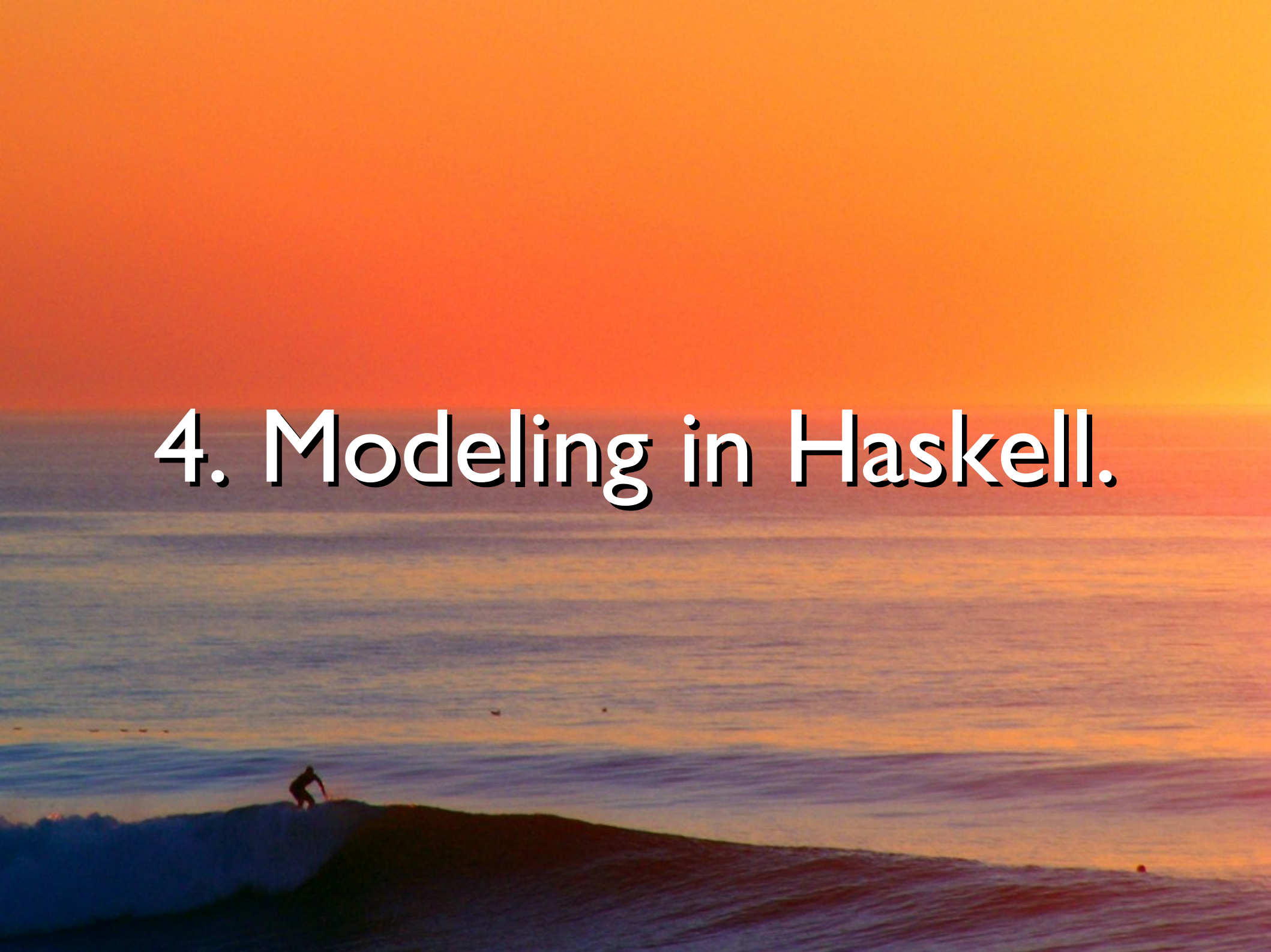

You need:
Tenacity.
An idea.
Tenacity.
Explanatory skill.
Tenacity.

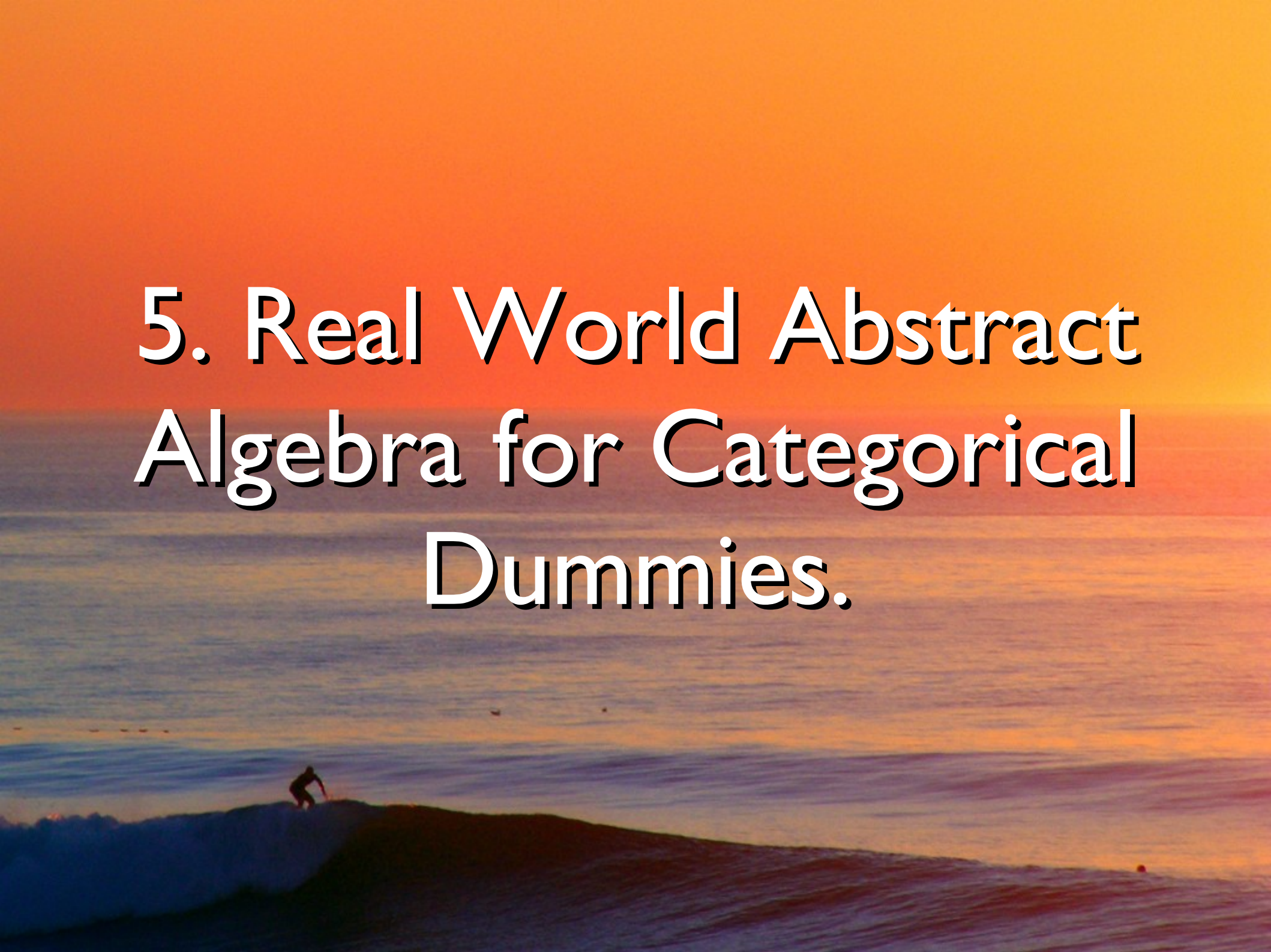

I even have
some ideas
for you.
All nice and handy.


1. Writing and Tuning Fast
Haskell Code.


2. Parallel Haskell in 24 Hours.

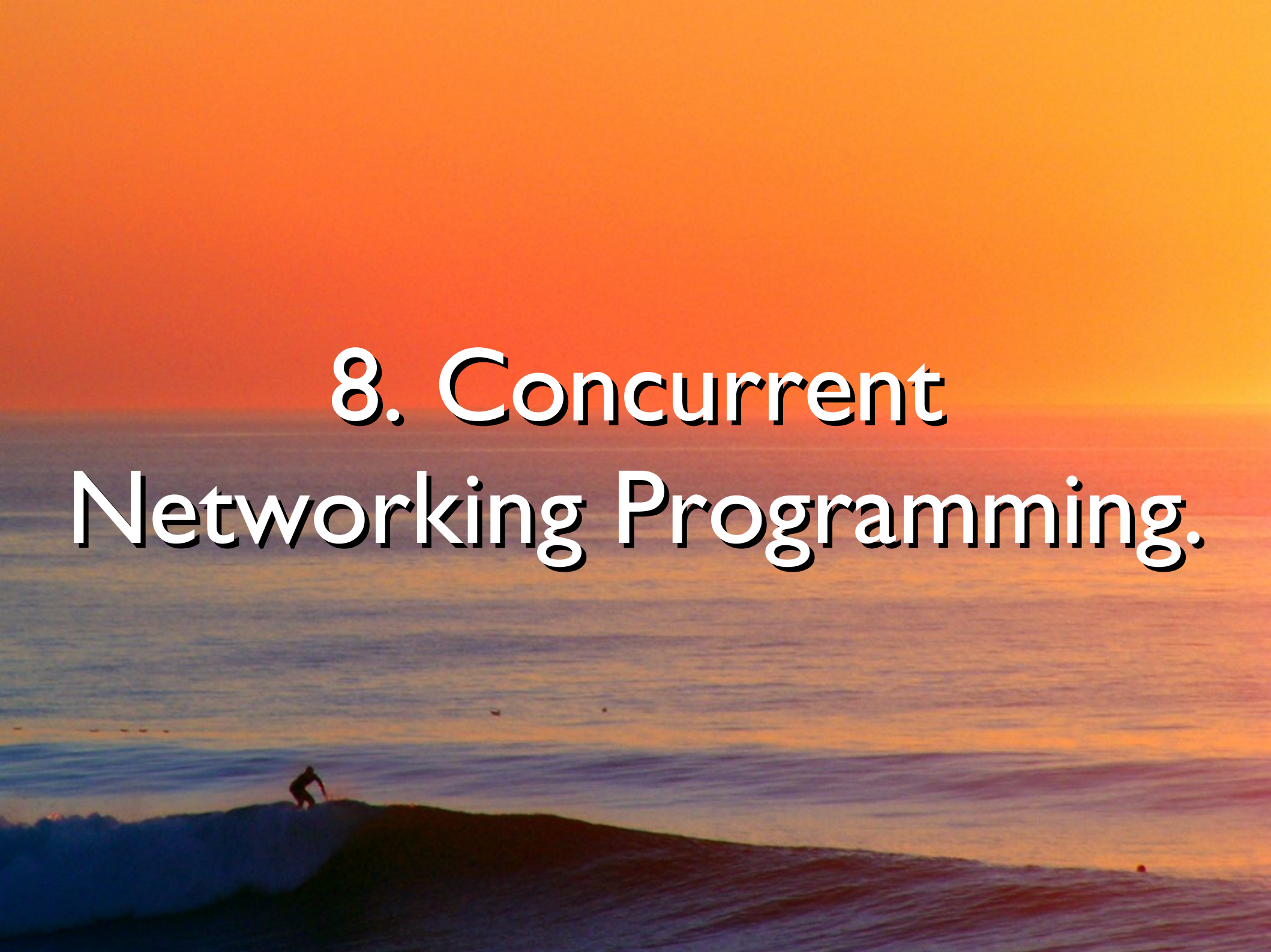
3. High-Assurance Haskell.


4. Modeling in Haskell.


5. Real World Abstract Algebra for Categorical Dummies.

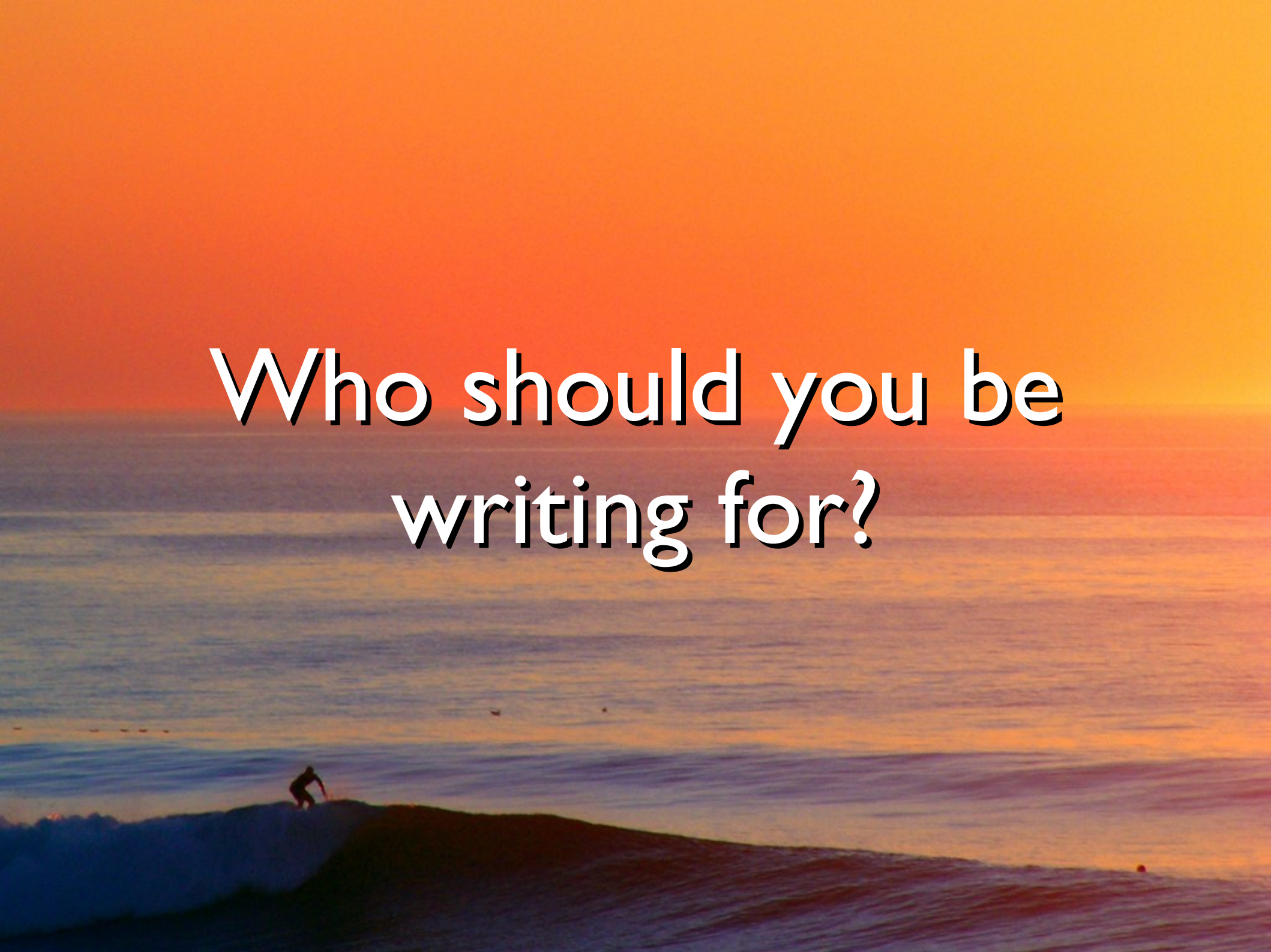
6. Complex Functional
Applications.

7. Grow Your Own
Monads in the Basement,
Legally.


8. Concurrent
Networking Programming.

9. Functional Pearls and
Design Patterns.

10. Secrets of the
QuickCheck Wizards.

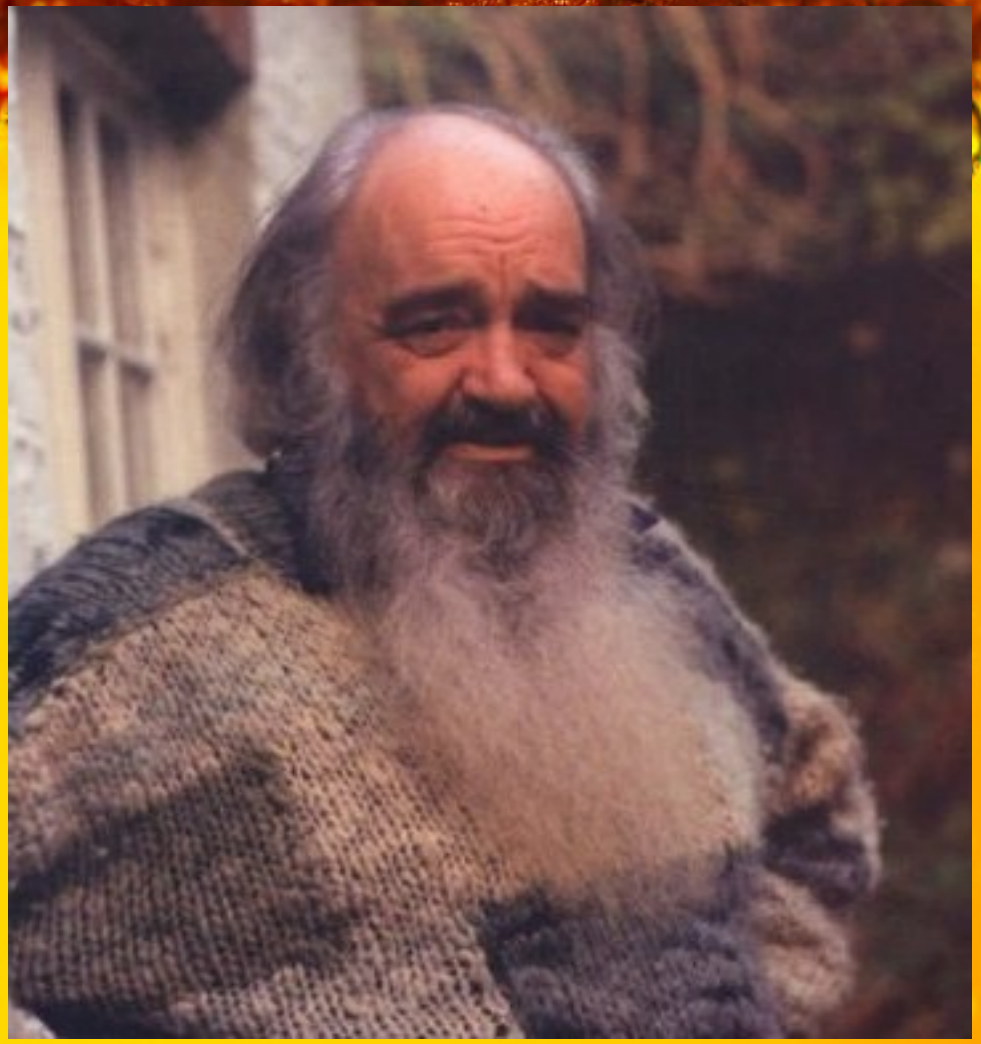

Who should you be writing for?

NOT THE PEOPLE
IN THIS ROOM.

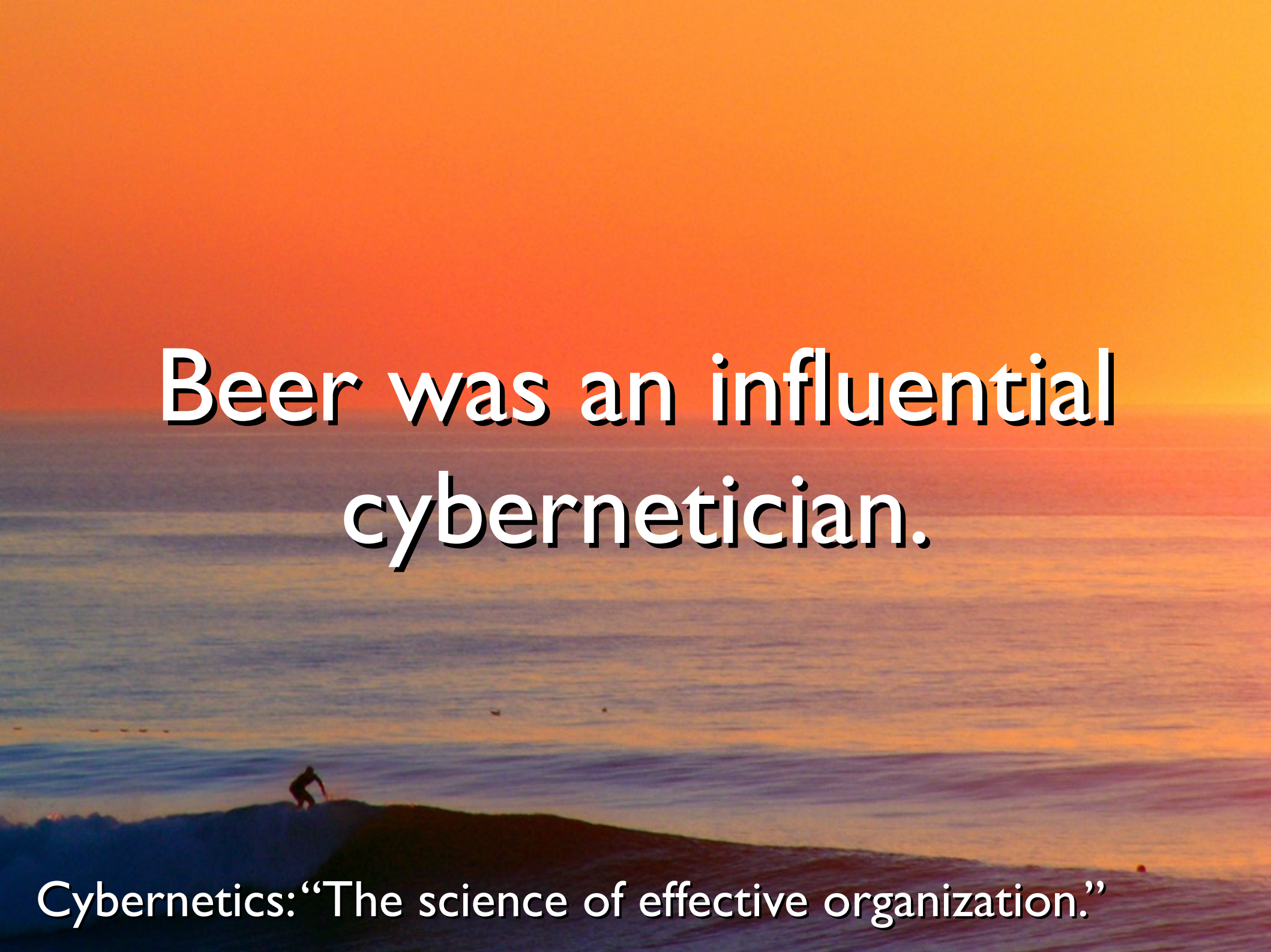
Look to
other programming
communities.

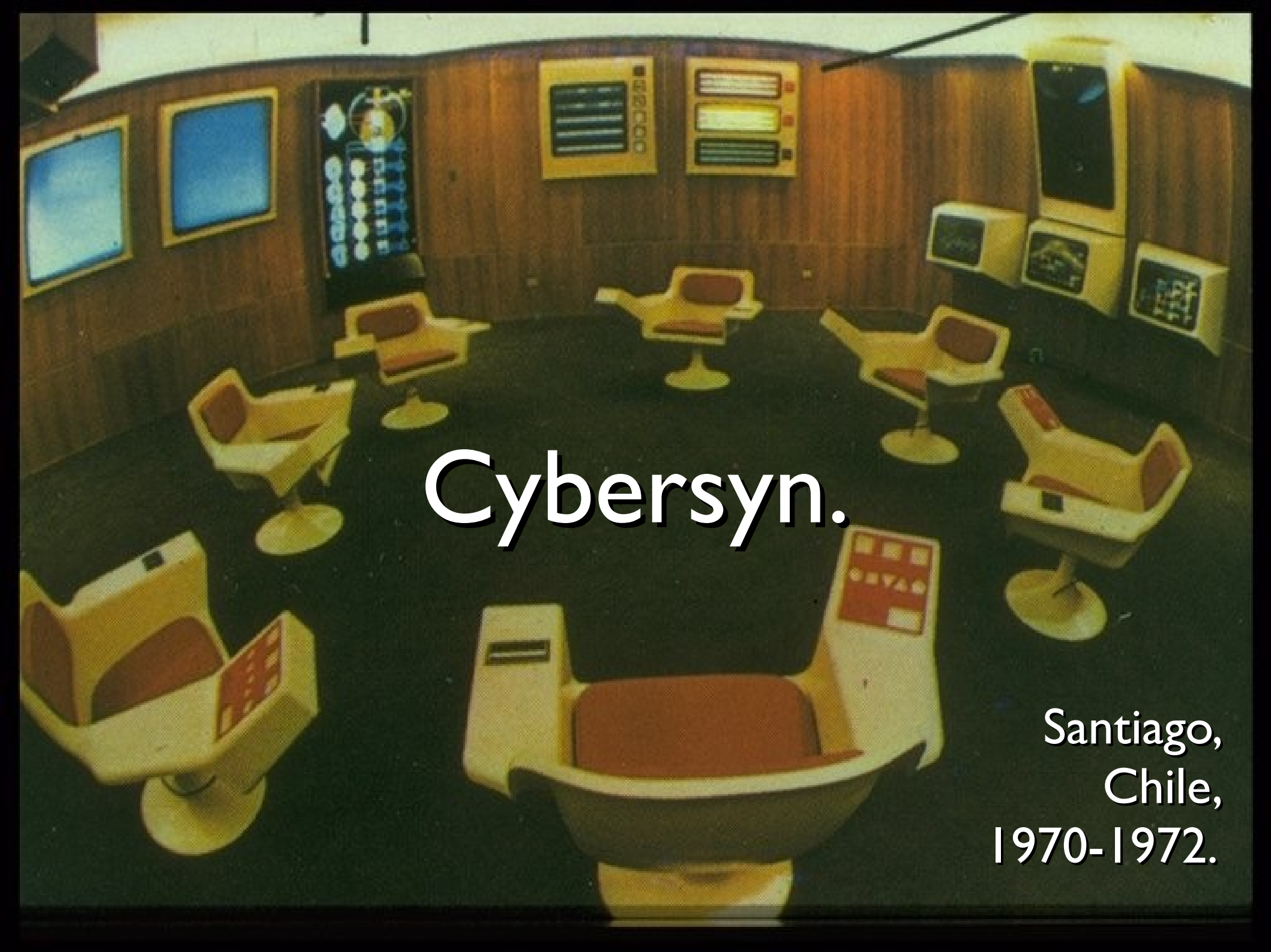
If we don't talk
to them, they
won't simply
come to us.

Who is this man?



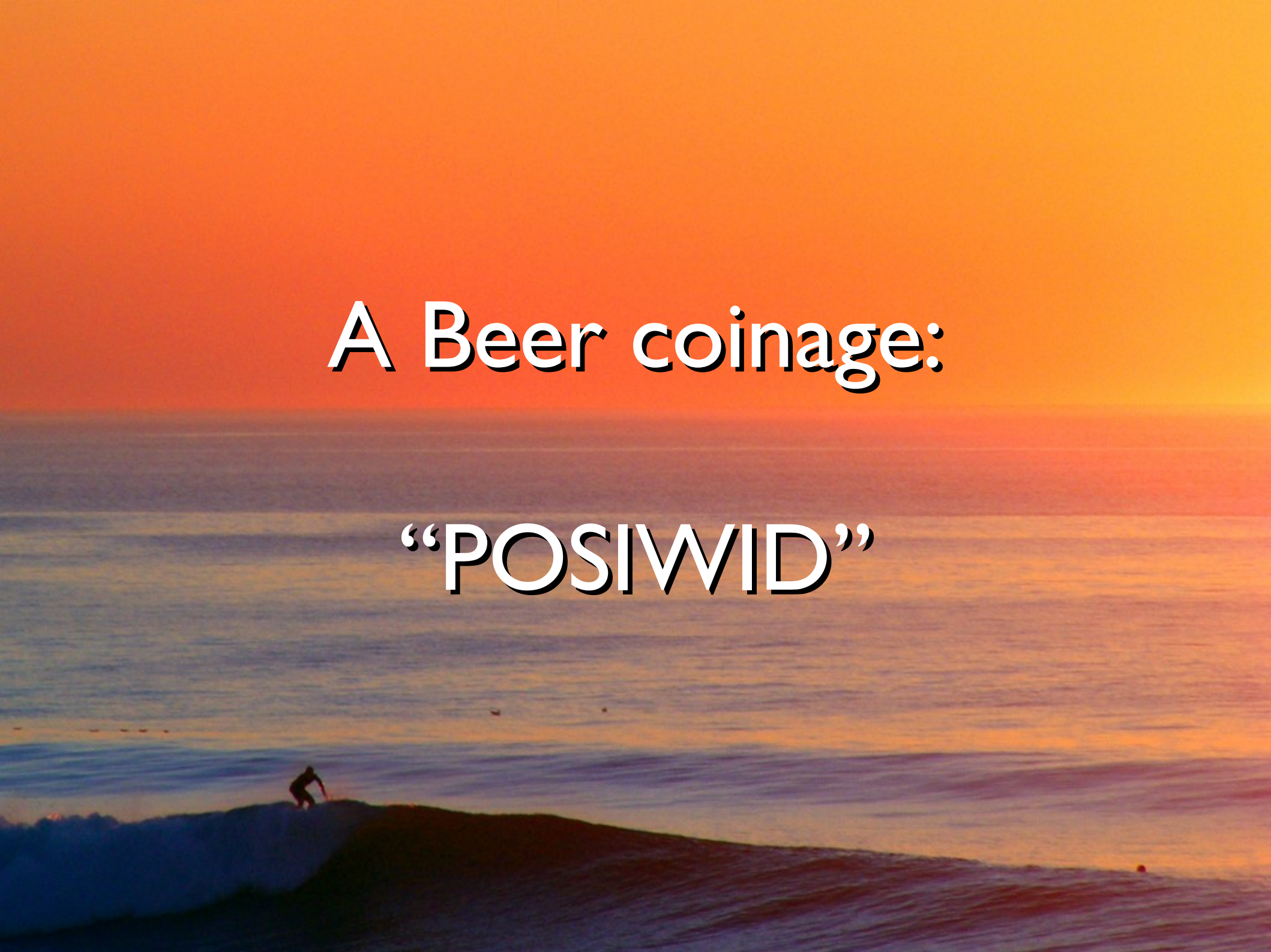
Anthony Stafford Beer.

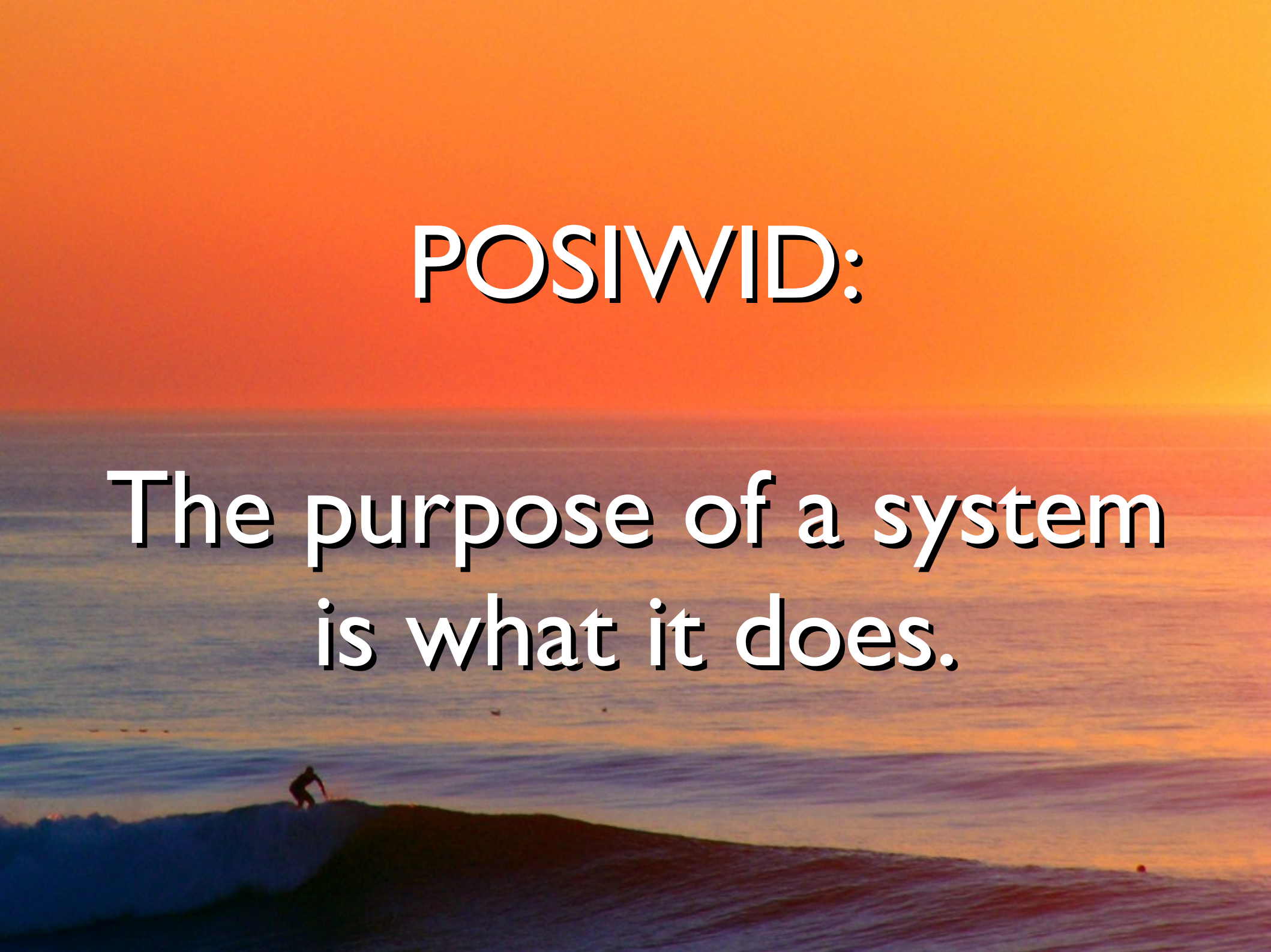

Beer was an influential cybernetician.
Cybernetics: “The science of effective organization.”

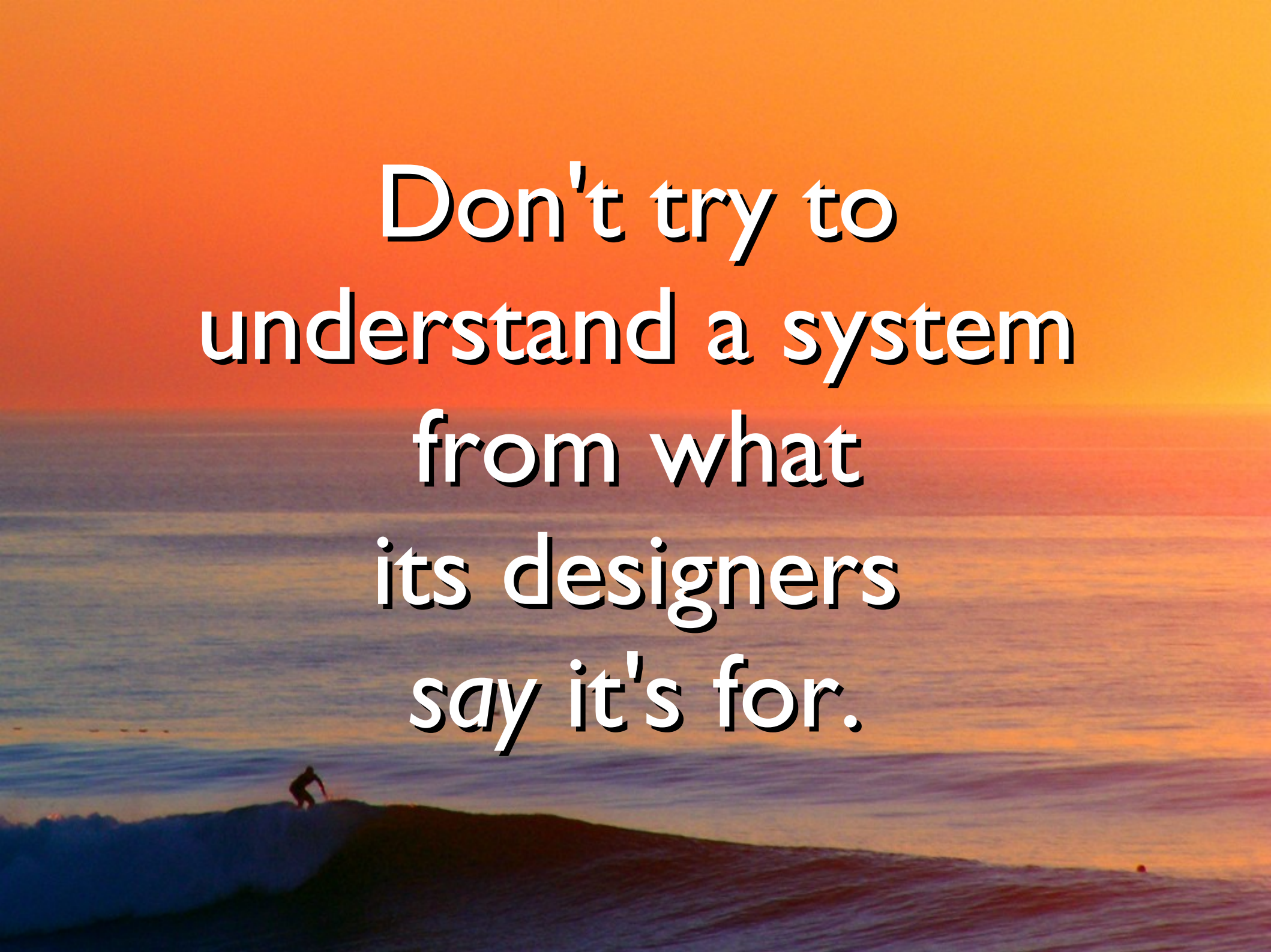

Cybersyn.
Santiago,
Chile,
1970-1972.

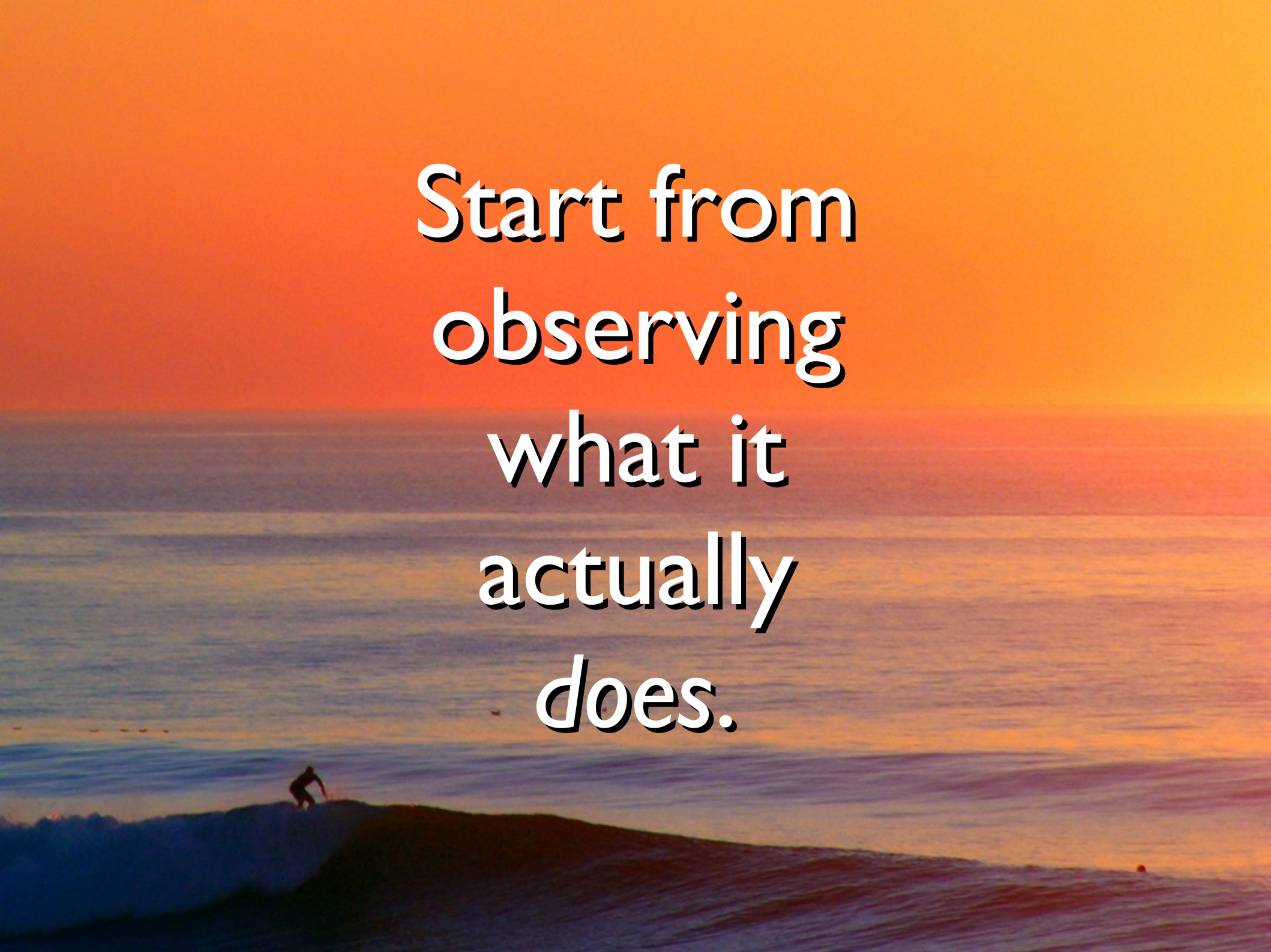
Cybersyn:
Centralised control of the Chilean economy.

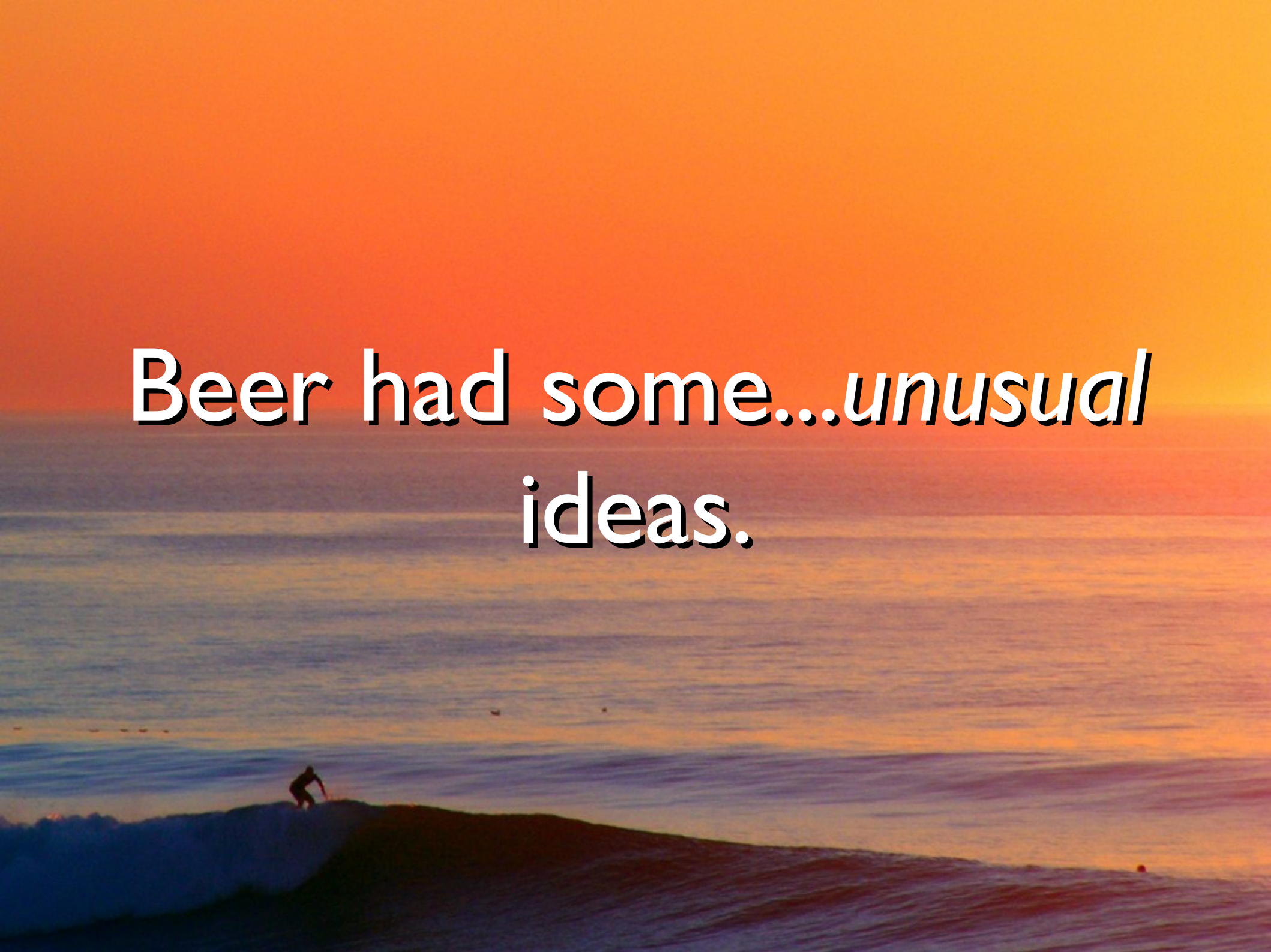
A star network of telex machines running from factories to a control centre in Santiago.

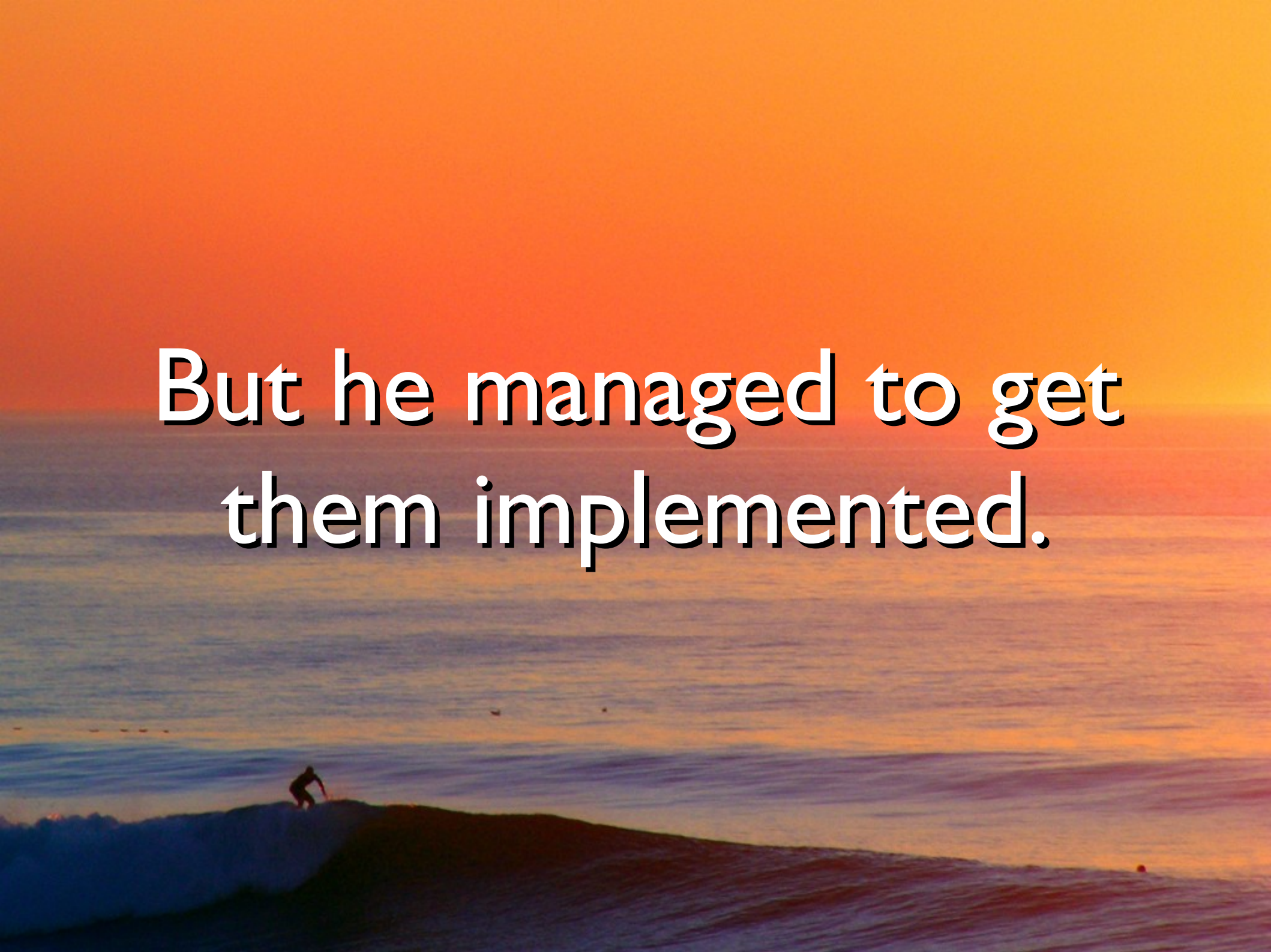

A Beer coinage:
“POSIWID”

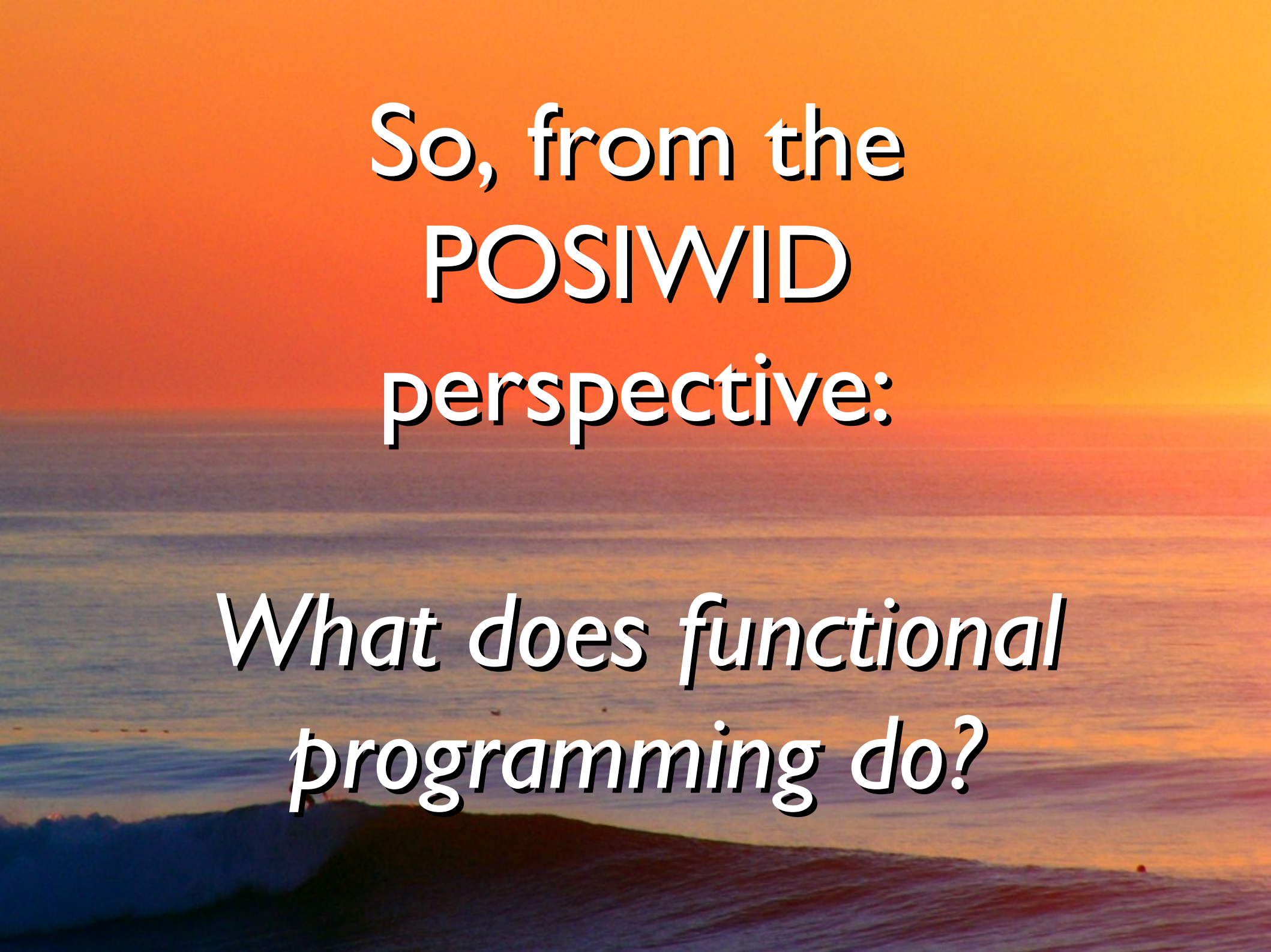

POSIWID:
The purpose of a system
is what it does.

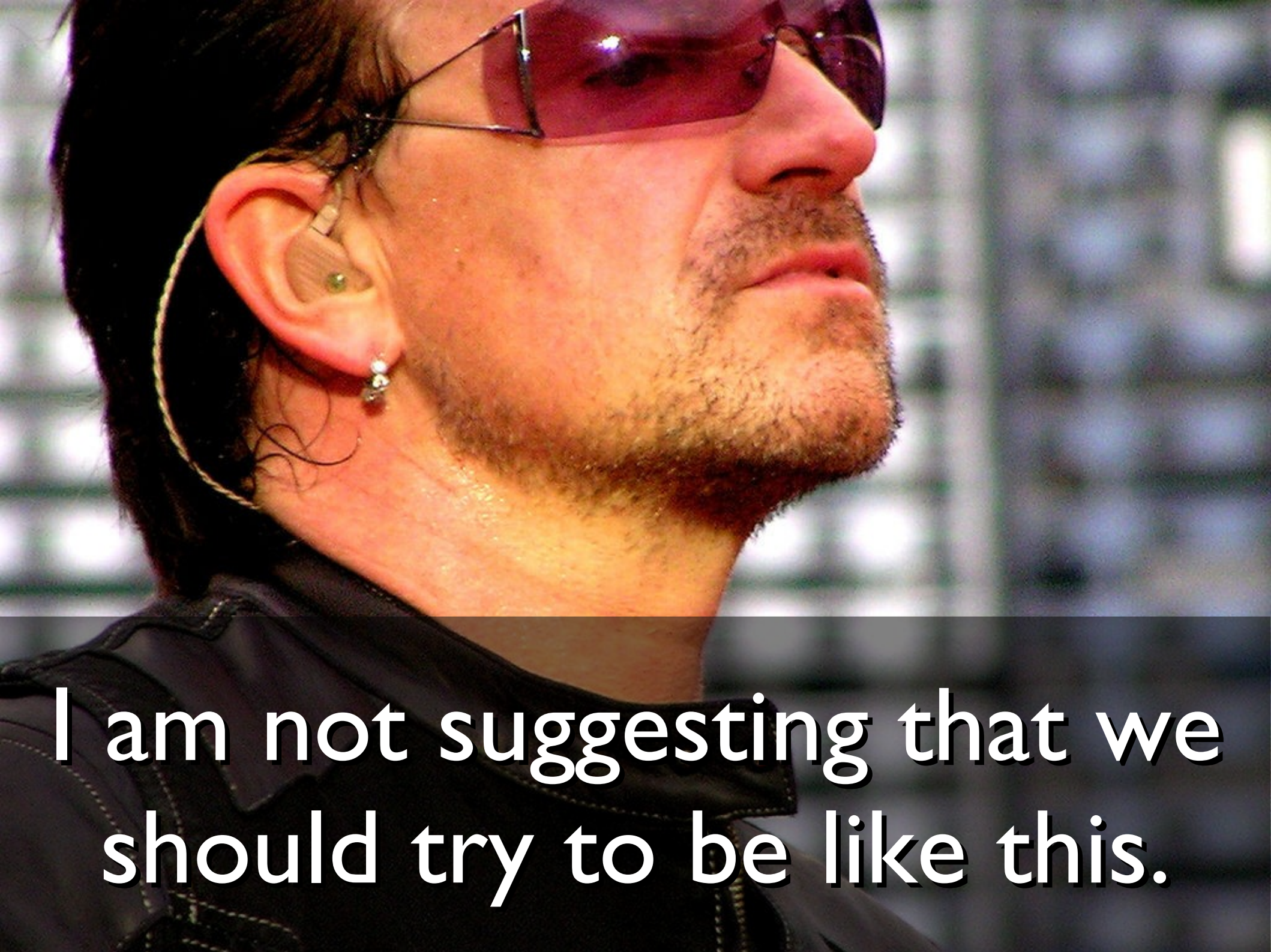

Don't try to
understand a system
from what
its designers
say it's for.

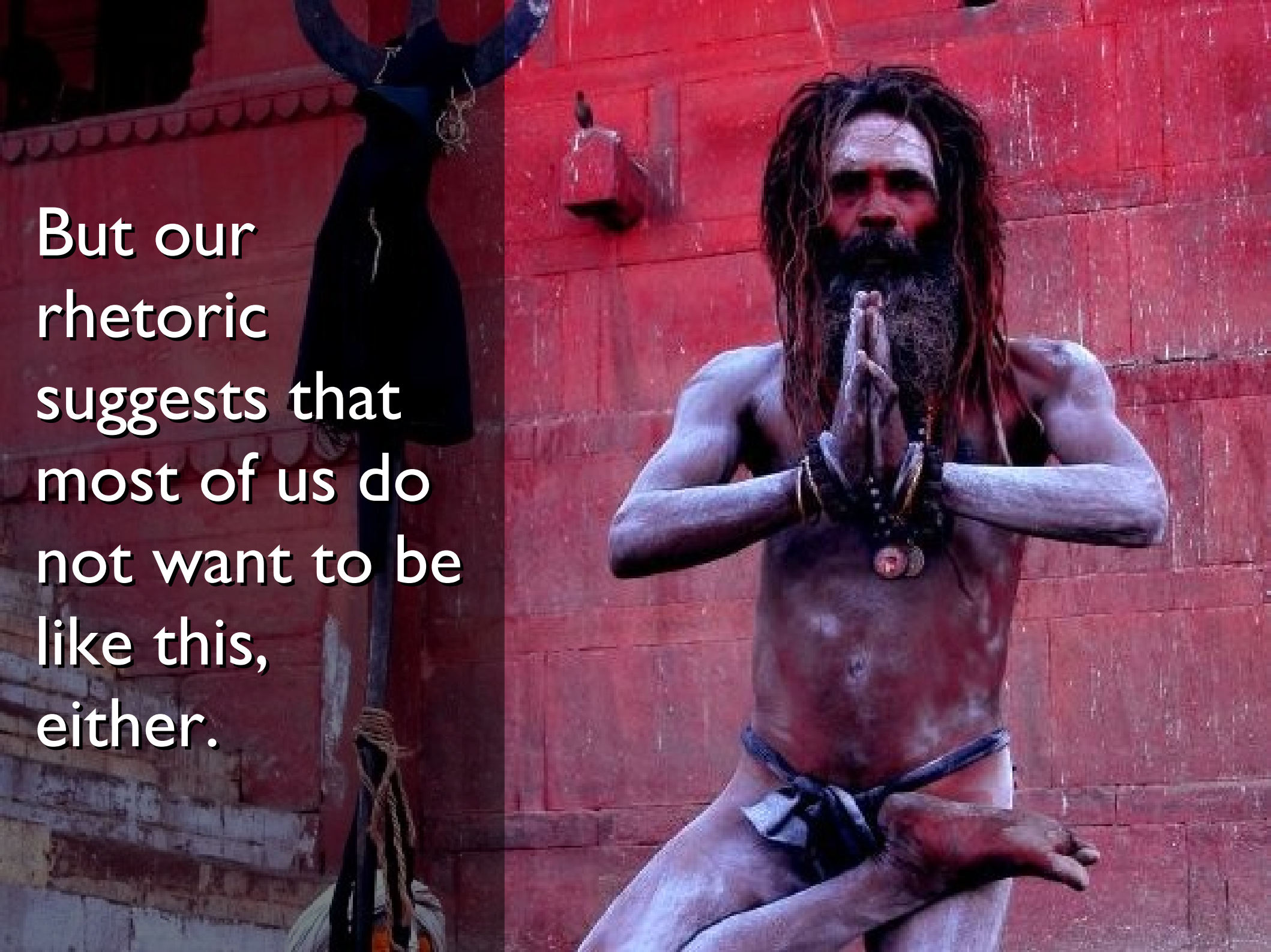

Start from
observing
what it
actually
does.


Beer had some...*unusual*
ideas.


But he managed to get
them implemented.


So, from the POSIWID perspective:
What does functional programming do?


I am not suggesting that we
should try to be like this.


But our rhetoric suggests that most of us do not want to be like this, either.

Go write a book.
Give a talk.
Write a blog post.

Oh, and thanks.