xqerl

XQuery 3.1 Implementation in Erlang

Zachary N. Dean

XML Prague 2018

A little about me…

<u>Prior life</u>

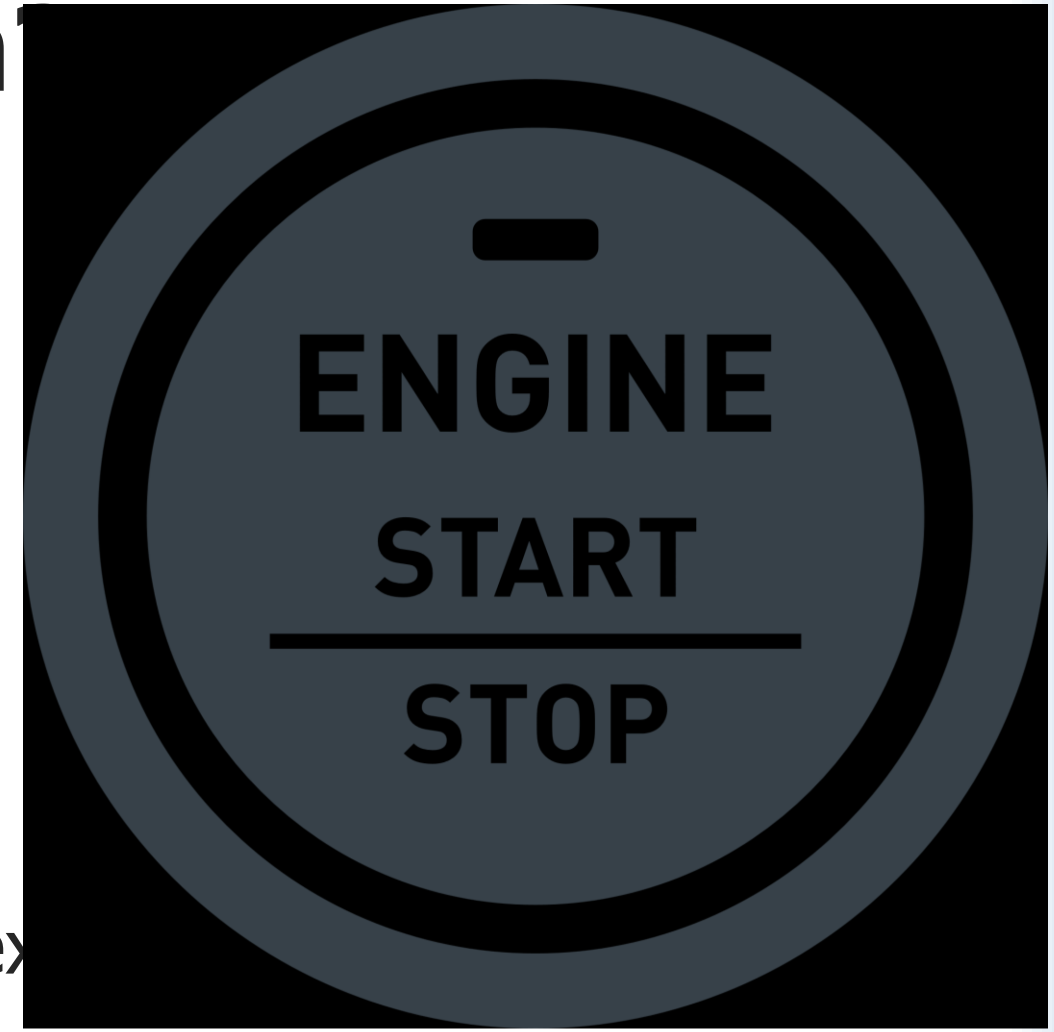
- Business Intelligence
- Relational Databases
- Data Warehouse
- Retail and Flight Industries

<u>Currently</u>

- Freelancer
- XQuery, Erlang, other

Where to begin

- Erlang?
- Why make xqerl?
- How does it work?
- Well, what can it do?
- Sounds cool, what's nex



Erlang?

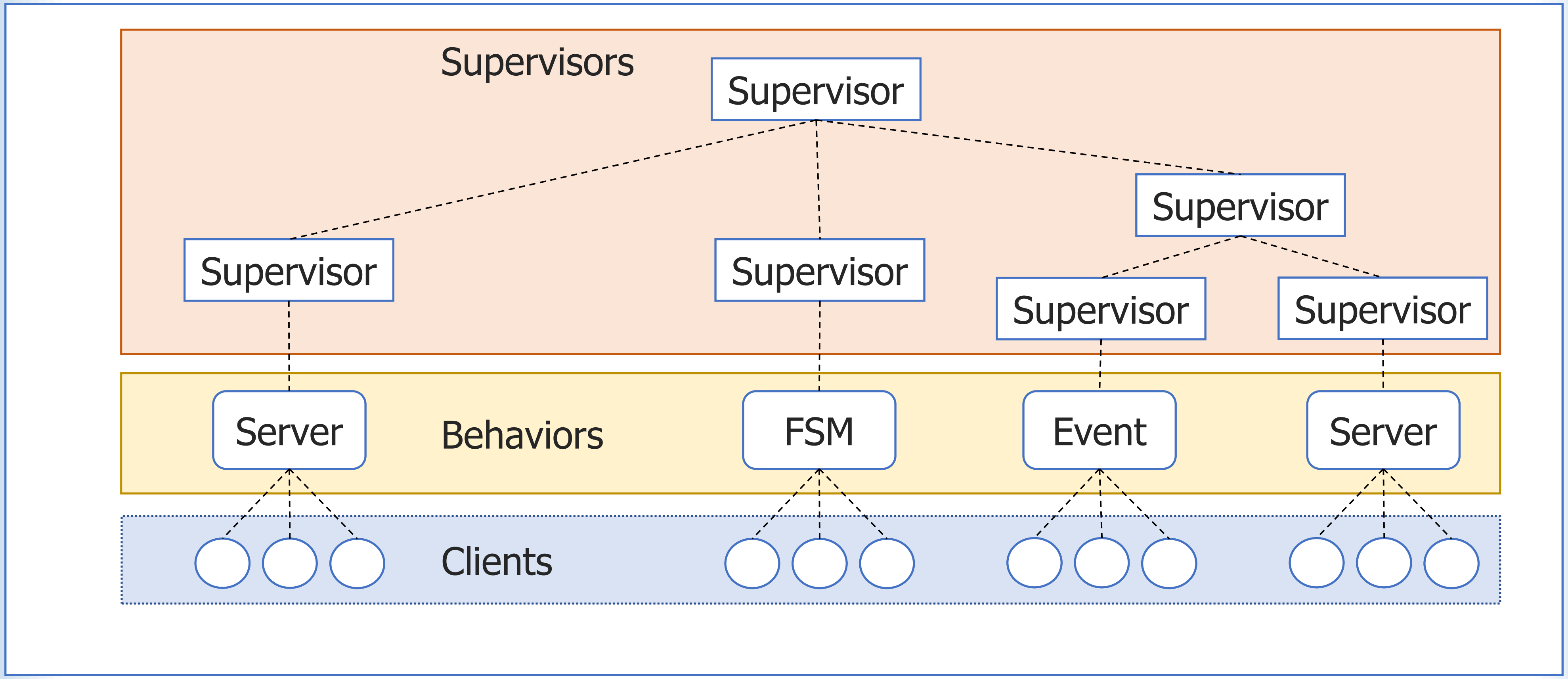
“Erlang is a programming language used to build massively scalable soft real-time systems with requirements on high availability. Some of its uses are in telecoms, banking, e-commerce, computer telephony and instant messaging. Erlang's runtime system has built-in support for concurrency, distribution and fault tolerance.”

http://www.erlang.org/

Erlang?

- Ericsson Computer Science Laboratory in 1986
- Open Source in 1998
- Functional Language
- Concurrency Oriented
- Actor Model
- Lightweight Processes
- Asynchronous Message Passing
- Shared Nothing

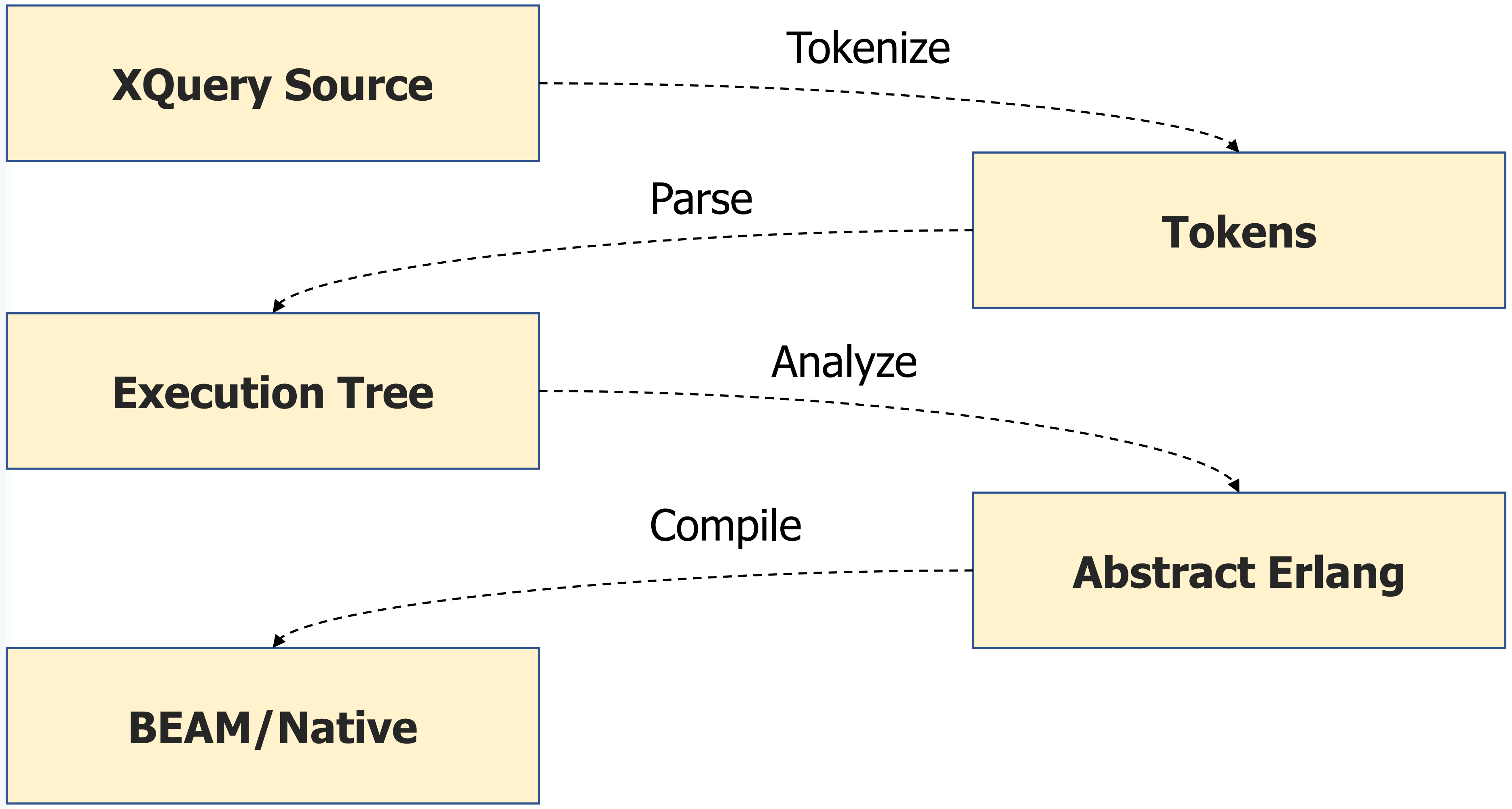
Supervision Tree



Why make xqerl?

- Was between projects
- Looking for something new
- Wanted to use Erlang
- Already liked using XQuery
- A few beers later…

How does it work?



How does it work?

```
for $x in (2,1,3)
let $y := -$x
order by $y
return $x
```

How does it work?

```
main() ->
   VarTup__1 = for__1(new),
   VarTup__2 = xqerl_flwor:orderbyclause(
                 VarTup__1,
                 [{fun ({_XQ__var_1, XQ__var_2}) ->
                          XQ__var_2
                   end,
                   ascending, greatest}]),
   return__3(VarTup__2).

for__1(new) ->
    List = [{xqAtomicValue, 'xs:integer', 2},
            {xqAtomicValue, 'xs:integer', 1},
            {xqAtomicValue, 'xs:integer', 3}],
    for__1(List);
for__1([]) -> [];
for__1([XQ__var_1 | T]) -> [let__2({XQ__var_1}) | for__1(T)].

let__2({XQ__var_1}) ->
    XQ__var_2 = xqerl_operators:unary_minus(XQ__var_1),
    {XQ__var_1, XQ__var_2}.

return__3(List) when is_list(List) -> [return__3(T) || T <- List];
return__3({XQ__var_1, _XQ__var_2}) -> XQ__var_1.
```

What can it do?

So far…

- Passes around 99% of QT3 tests run
- Higher-Order Function Feature
- Module Feature
- UCA 10.0

What can it not do yet?

<u>Optional Features</u>

- Schema Aware Feature
- Typed Data Feature
- Static Typing Feature
- Serialization Feature

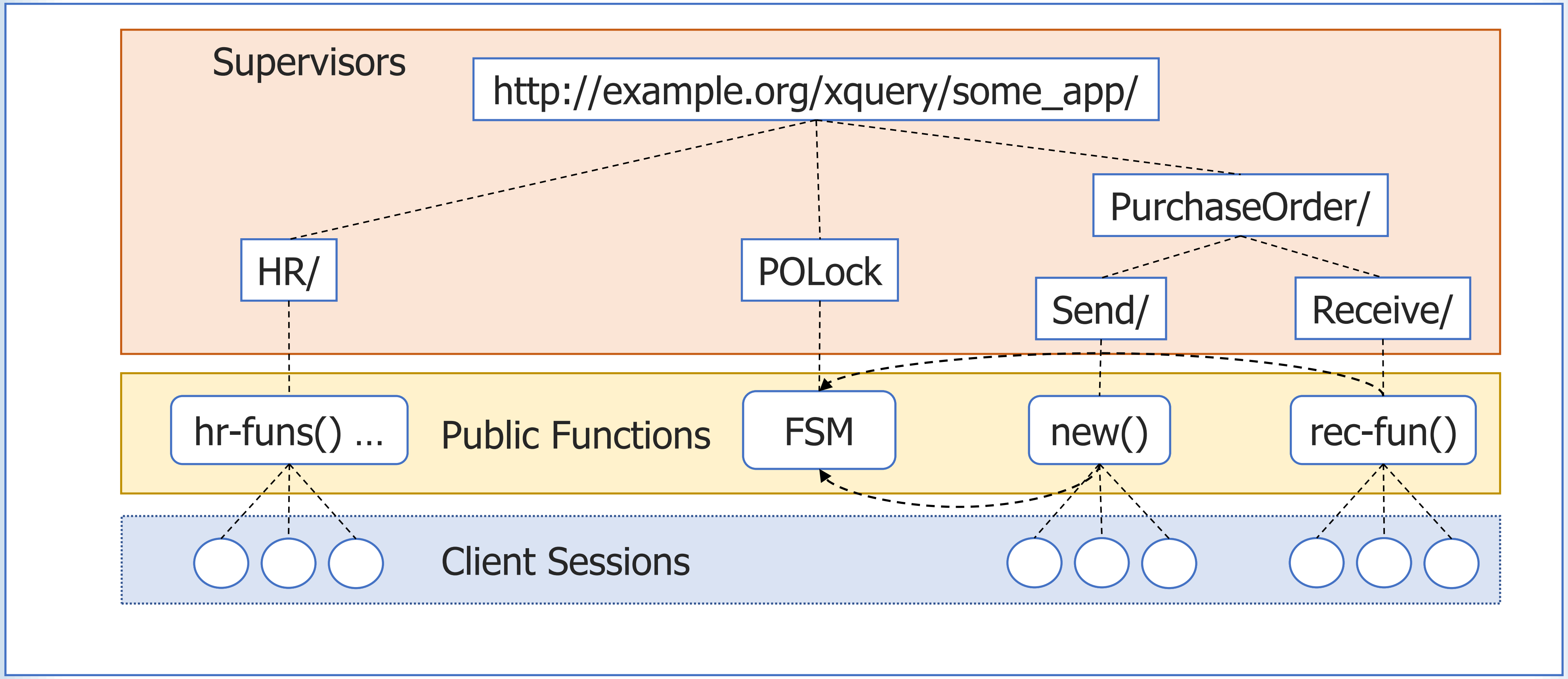
<u>Extras</u>

- Update Facility
- Full-Text

What’s next?

- Unimplemented optional features
- Cost-based parallel processing
- Bindings to external sources
  - NoSQL DBs
  - Relational DBs
- Websockets
- EXPath / EXQuery
  - RESTXQ
  - HTTP Client

XQuery Supervision Tree



Thank you!

contact@zadean.com

@ZacharyNDean

zadean/xqerl