



Writing A New Erlang/OTP Module for Beginners

Kenji Rikitake

23-MAR-2017

Erlang and Elixir Factory SF Bay 2017

San Francisco, CA, USA

@jj1bdx

- Erlang Factory SF Bay 2010-2016
speaker for seven times, and...
- Erlang and Elixir Factory SF Bay
2017 speaker (8th year!)
- Erlang rand module co-creator



what I did for OTP

**Wrote Erlang PRNG code
Tested and published the results
Put the code into OTP**

This talk is not about

Algorithms

Random numbers

Other implementation details

This talk is about
Development for Erlang/OTP
Working with OTP Team
Gaining support for your code

Development for Erlang/OTP

why new code?

**Bugfix/security
New features**

"Fix what I don't like"

**Do you have to
change OTP?**

**OTP is for *all* Erlang users
Who needs new code?**

Once committed, removal is hard

Rationale for rand module

- Period too short
- Better API (no initialization)
- Multiple algorithms
- Other features

Prototyping Independent repository Common Test and Dialyzer Learn erl_docgen

Common Test and parallelism

```
all() -> % From OTP master lib/stdlib/test/rand_SUITE.erl
    [seed, interval_int, interval_float, api_eq, reference,
     {group, basic_stats},
     plugin, measure,
     {group, reference_jump}].

groups() ->
    [{basic_stats, [parallel],
      [basic_stats_uniform_1, basic_stats_uniform_2,
       basic_stats_normal]},
     {reference_jump, [parallel],
      [reference_jump_state, reference_jump_procdict]}].
```

Improper lists and Dialyzer

```
%%% From OTP master lib/stdlib/src/rand.erl
%%% Directive to ignore improper list
-dialyzer({no_improper_lists, exsplus_next/1}).

-type exsplus_state() :: nonempty_improper_list(uint58(), uint58()).

-spec exsplus_next(exsplus_state()) -> {uint58(), exsplus_state()}.

exsplus_next([S1 | S0]) ->
    %% Note: members s0 and s1 are swapped here
    S11 = (S1 bxor (S1 bsl 24)) band ?UINT58MASK,
    S12 = S11 bxor S0 bxor (S11 bsr 11) bxor (S0 bsr 41),
    {(S0 + S12) band ?UINT58MASK, [S0|S12]}.
```

erl_docgen document example

```
<func>
  <name name="jump" arity="0"/>
  <fsummary>Return the seed after performing jump calculation
    to the state in the process dictionary.</fsummary>
  <desc><marker id="jump-0" />
    <p>Returns the state
      after performing jump calculation
      to the state in the process dictionary.</p>
    <p>This function generates a <c>not_implemented</c> error exception
      when the jump function is not implemented for
      the algorithm specified in the state
      in the process dictionary.</p>
  </desc>
</func>
```

< > C Home file:///Users/kenji/txt/writing-otp-modules/draft/source/_static/rand.html ▾ ▾ Search Google

ERL

ALANG

PDF Top +

Reference Manual Version 0.9.2

Expand All Contract All

Table of Contents

`export_state()`

Algorithm-dependent state that can be printed or saved to file.

EXPORTS

`export_seed() -> undefined | export_state()`

Returns the random number state in an external format. To be used with `seed/1`.

`export_seed_s(X1 :: state()) -> export_state()`

Returns the random number generator state in an external format. To be used with `seed/1`.

`jump() -> NewS :: state()`

Returns the state after performing jump calculation to the state in the process dictionary.

This function generates a `not_implemented` error exception when the jump function is not implemented for the algorithm specified in the state in the process dictionary.

`jump(State :: state()) -> NewS :: state()`

Returns the state after performing jump calculation to the given state.

This function generates a `not_implemented` error exception when the jump function is not implemented for the algorithm specified in the state.

`normal() -> float()`

Returns a standard normal deviate float (that is, the mean is 0 and the standard deviation is 1) and updates the state in the process dictionary.

`normal_s(State0 :: state()) -> {float(), NewS :: state()}`

Returns, for a specified state, a standard normal deviate float (that is, the mean is 0 and the standard deviation is 1) and a new state.

seed(AlgorithmState :: state() | export_state()) -> state()



Working with OTP Team: Advice from the author of Cowboy

Tweet from Loïc Hoguin (@lhoguin at Twitter)



Loïc Hoguin
@lhoguin

[!\[\]\(81312b19ca3202a7c3e2f42667ac19f0_img.jpg\) Follow](#)

How not to get changes merged:

- Send big patch without previous discussion
- Get patch rejected with request for changes
- Not doing those

5:55 AM - 24 Feb 2017

← 1 ↗ 2

Tweet from Loïc Hoguin (@lhoguin at Twitter)



Loïc Hoguin
@lhoguin

[!\[\]\(f9640f02d06b604c50c1e3a558a31388_img.jpg\) Follow](#)

How to get changes merged:

- Discuss big picture with maintainer
- Write the changes one small step at a time
- Submit them in small chunks

5:56 AM - 24 Feb 2017

↳ 1 ❤

Tweet from Loïc Hoguin (@lhoguin at Twitter)



Loïc Hoguin
@lhoguin

The key is the constant discussion around the changes.

5:57 AM - 24 Feb 2017

1 2

[Follow](#)

A screenshot of a tweet from Loïc Hoguin (@lhoguin) on Twitter. The tweet contains the text "The key is the constant discussion around the changes." It was posted at 5:57 AM on February 24, 2017. The tweet has received 1 reply and 2 likes. A "Follow" button is visible on the right side of the tweet card.

what to include in a GitHub PR

**What and how to do
What will be affected
Test cases and type specs
Documentation**

How to issue a PR

Report bugs *before* PR

Choose right branch

Separate commits for separate changes

Make sure each commit works

Communication with OTP Team

Use private emails if needed

OTP Team work in Sweden Time

OTP Team have their weekends

OTP Team have to handle many modules

A photograph of a group of people gathered around a campfire at night. A man in the foreground on the left is wearing a hat and a plaid shirt, smiling. To his right, a woman with long hair is also smiling. In the background, another person's face is partially visible. The scene is lit by the fire, creating a warm atmosphere.

How to gain community support for your code

Promote your code

Publish on GitHub and elsewhere

Give talks at conferences

Write PoC and show the problems

Code maintenance and support

**You are responsible for your code
Further contribution expected
Old code may be removed
Be careful on adding features**

Erlang/OTP License: Apache 2

incompatible with GPLv2/v3
MIT/BSD code: ok to merge
Relicensing often needed

A collage of three men playing guitars. On the left, a man wearing sunglasses and a cap plays an acoustic guitar. In the center, a man with a tattooed arm and a cap plays a banjo. On the right, a man wearing a cap and a dark shirt plays a double bass. The background is dark and textured.

**OTP needs your help
Your contribution is always welcome**

References

- My WIP document: Writing OTP Modules
 - <http://docs.jj1bdx.tokyo/writing-otp-modules/html/index.html>
- Source of Writing OTP Modules
 - <https://github.com/jj1bdx/writing/otp-modules>

Support for this presentation is provided by



Pepabo R&D Institute, GMO Pepabo, Inc.

Acknowledgment

- Dan Gudmundsson - rand module principal developer
- Sebastiano Vigna - Xorshift*/+ inventor
- Erlang Solutions



thank you
questions?

Photo credits:

- Title slide: Davide Ragusa, from Unsplash.com
- Kenji Rikitake's face: Yutaka Sakurai and Naoki Sakurai, taken in front of USS Pampanito at Pier 45, San Francisco, CA, USA, March 2015
- "This talk is not about" slide: Markus Spiske, from Unsplash.com
- "Development for Erlang/OTP" slide: Luis Llerena, from Unsplash.com
- "Prototyping" slide: Bram Naus, from Unsplash.com
- "Working with OTP Team" slide: Johann Walter Bantz, from Unsplash.com
- "How to gain community support" slide: Clem Onojeghuo, from Unsplash.com
- "OTP needs your help" slide: Matheus Ferrero, from Unsplash.com
- "Thank you" slide: Chris Brignola, from Unsplash.com

(All Unsplash.com photos are licensed under Creative Commons CC0 License)