

# Introducing Erlang

分享者：mingchaoyan@gmail.com

# Who am I ?

- Erlang , PHP / 《明朝这些年》
- Erlang / C++ , Lua , Cocos2d / 《全萌猎人》
- Erlang / C# , Lua , Unity , uLua / 《勇者联盟》
- JavaScript (CoffeeScript) , Node.js / 《谁是卧底OL》
- Ruby , RoR / 《聚会玩平台管理系统》
- C# , Unity / C , Lua , Skynet / 《消消果缤纷》

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型
- 重要的语法现象
- 没有讲到的话题
- 推荐
- 小结
- Q / A

# 大纲

- hello, world

# hello, world

## Erlang shell

```
$ erl
Erlang/OTP 19 [erts-8.1] [source] [64-bit] [smp:4:4] [async-threads:10] [hipe] [kerne
Eshell V8.1 (abort with ^G)
1> io:format("hello, world\n").
hello, world
ok
2> q().
ok
3> $
```

# hello, world

## Script

- 编辑

```
#! /usr/bin/env escript
main(_) ->
    io:format("hello, world\n").
```

- 解释运行

```
$ chmod +x hello.erl
$ ./hello.erl
hello, world
```

# hello, world

## Compile

- 编辑

```
-module(hello).  
-export([hello_world/0]).  
hello_world() ->  
    io:fwrite("hello, world\n").
```

- 编译产生beam文件

```
$ erlc hello.erl  
$ ls  
$ hello.beam
```

- 运行erl 载入beam文件

```
$ erl  
> hello:hello_world().  
> hello, world
```

# 大纲

- hello, world
- 历史 & 作者

# 历史

- 出现: 1986
- 开源发布: 1998
- 作者: 乔·阿姆斯特朗 (瑞典) , 罗伯·维丁 (瑞典) , 麦可·威廉 (瑞典)
- 设计初衷: 电信设备制造商爱立信私有软件, 创造一种可以应付大规模并发活动的程序设计语言和运行环境

# 作者



Robert Virding

Creator of Erlang

Mike Williams

Creator of Erlang

Joe Armstrong

Creator of Erlang

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性

# 编程范型 & 语言特性

- 函数式
- 并发
- 消息通信
- 容错/热更新

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型

# 数据类型（动态强类型）

## 基本类型

- 原子(Atom): foo, bar, true, false
- 整数(Integer):42
- 浮点数(Float):3.14
- 二进制(Bit/Binary): <<10, 20>>

## 基本类型 (续)

- 函数 (Func)
- A fun is a functional object.

```
1> Fun1 = fun (X) -> X+1 end.  
#Fun<erl_eval.6.39074546>  
2> Fun1(2).  
3
```

- 一等公民 (first class)

## 基本类型(续)

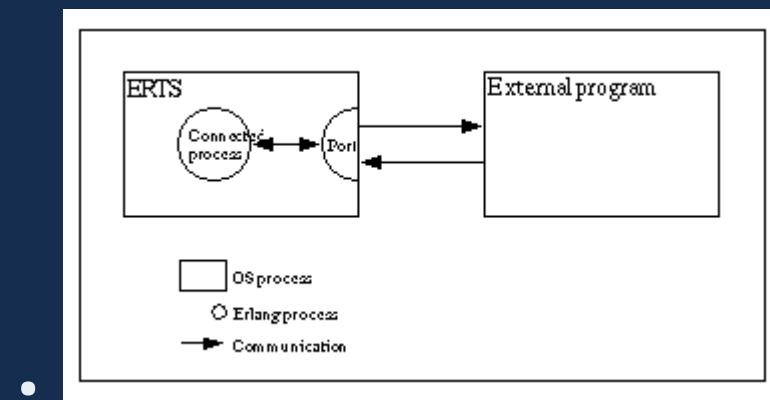
- 进程ID (Pid)
- A process identifier, pid, identifies a process.

```
1> spawn(m, f, []).  
<0.51.0>
```

- Erlang VM级别的进程，并非OS级别进程
- Actor并发模型的基础，Skynet受其启发

## 基本类型（续）

- 端口 (Port)
- A port identifier identifies an Erlang port.



## 基本类型（续）

- 引用 (Ref)
- A reference is a term that is unique in an Erlang runtime system

```
> make_ref().  
#Ref<0.0.4.99>
```

## 复合类型

- List / String
- A list is a compound data type with a variable number of terms.

```
1> L1 = [a,2,{c,4}].  
[a,2,{c,4}]  
2> [H|T] = L1.  
[a,2,{c,4}]  
3> H.  
a  
4> T.  
[2,{c,4}]  
5> L2 = [d|T].  
[d,2,{c,4}]  
6> length(L1).  
3  
7> length([]).  
0
```

## 复合类型（续）

- 元组 (Tuple)
- A tuple is a compound data type with a fixed number of terms:

```
1> P = {adam,24,{july,29}}.
{adam,24,{july,29}}
2> element(1,P).
adam
3> element(3,P).
{july,29}
4> P2 = setelement(2,P,25).
{adam,25,{july,29}}
5> tuple_size(P).
3
6> tuple_size({}).
0
```

## 复合类型（续）

- 记录 (Record)
- A record is a data structure for storing a fixed number of elements.
- Record 是 Tuple 的一种语法糖

```
1 -module(person).  
2 -export([new/2]).  
3 -record(person, {name, age}).  
4 new(Name, Age) ->  
5     #person{name=Name, age=Age}.
```

```
1> person:new(ernie, 44).  
{person,ernie,44}
```

## 复合类型（续）

- 映射 (Map)
- Maps are considered to be experimental during Erlang/OTP

R17

```
1> M1 = #{name=>adam,age=>24,date=>{july,29}}.  
#{age => 24,date => {july,29},name => adam}  
2> maps:get(name,M1).  
adam  
3> maps:get(date,M1).  
{july,29}  
4> M2 = maps:update(age,25,M1).  
#{age => 25,date => {july,29},name => adam}  
5> map_size(M).  
3  
6> map_size(#{}).  
0
```

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型
- **重要的语法现象**

# 重要的语法现象

## 单一赋值

- Single Assignment

```
1> X = 1.  
1  
2> X = 2.  
** exception error: no match of right hand side value 2
```

- 没有竞态 (Race Condition) , 没有锁!
- 没有可变状态, 引用透明 (Referential Transparency) , 没有副作用 (No Side Effect)
- debug 方便

# 模式匹配

- Pattern Match

```
1> X.  
** 1: variable 'X' is unbound **  
2> X = 2.  
2  
3> X + 1.  
3  
4> {X, Y} = {1, 2}.  
** exception error: no match of right hand side value {1,2}  
5> {X, Y} = {2, 3}.  
{2,3}  
6> Y.  
3
```

## 尾递归

- Erlang 严重依赖递归
- 普通递归

```
duplicate(0, _) -> [];
duplicate(N, T)  when N > 0 -> [T | duplicate(N-1, T)].
```

- 尾递归 (Tail Recursive)

```
duplicate(0, _, L) -> L;
duplicate(N, X, L) -> duplicate(N-1, X, [X|L]).
```

- No Stack Overflow

# 热更新

- Erlang 设计初衷
- old vs current

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型
- 重要的语法现象
- **没有讲到的话题**

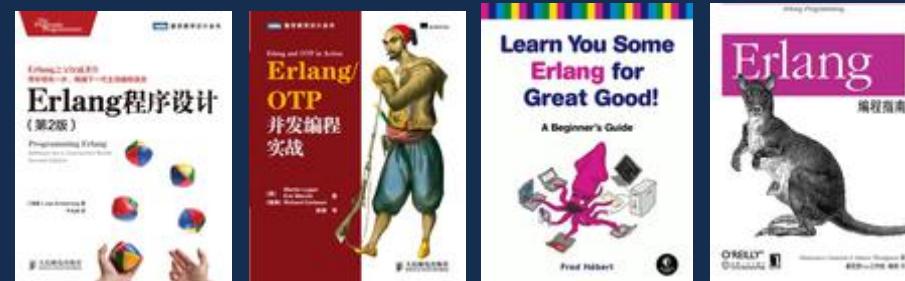
# 没讲到的话题

- 控制结构
- 错误处理
- 分布式
- ETS
- 并发原语／消息通信
- ...

# 大纲

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型
- 重要的语法现象
- 没有讲到的话题
- 推荐

# 推荐



- 余锋（淘宝褚霸）
- 成立涛
- 坚强2002

# 小结（大纲）

- hello, world
- 历史 & 作者
- 编程范型 & 语言特性
- 数据类型
- 重要的语法现象
- 没有讲到的话题
- 推荐
- 小结
- Q / A

Q / A

谢谢！