

elixir

Elixir is a dynamic, **functional** language designed for building **scalable** and maintainable applications.

В начале был...

Erlang

- Разработан компанией Ericsson в 1986 году
- Fault-tolerant, concurrent, distributed, scalable, real-time

Erlang

- Немутабельные данные
- Много легких процессов
- Нет доступа к «общим» данным
- Процессы общаются только через сообщения
- «Горячая» замена кода
- Принцип «пусть ломается» и 99,999999% uptime
- Проверенная платформа – Erlang VM (BEAM)

Erlang

- Amazon
- Yahoo!
- T-Mobile
- Motorola
- Heroku
- Call of Duty
- League of Legends
- Ericsson...

+

40%

TELEKOM

Erlang



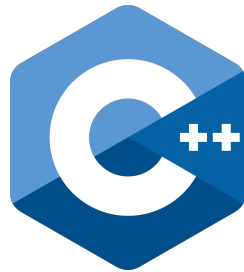
- 2 млн подключений на одном сервере
- 450 млн пользователей и 32 инженера
- 19 миллиардов USD заплатил Facebook

Два разных мира

Распределенные системы

- Надежность
- Миллионы пользователей
- Проверка временем

Erlang



Веб-приложения

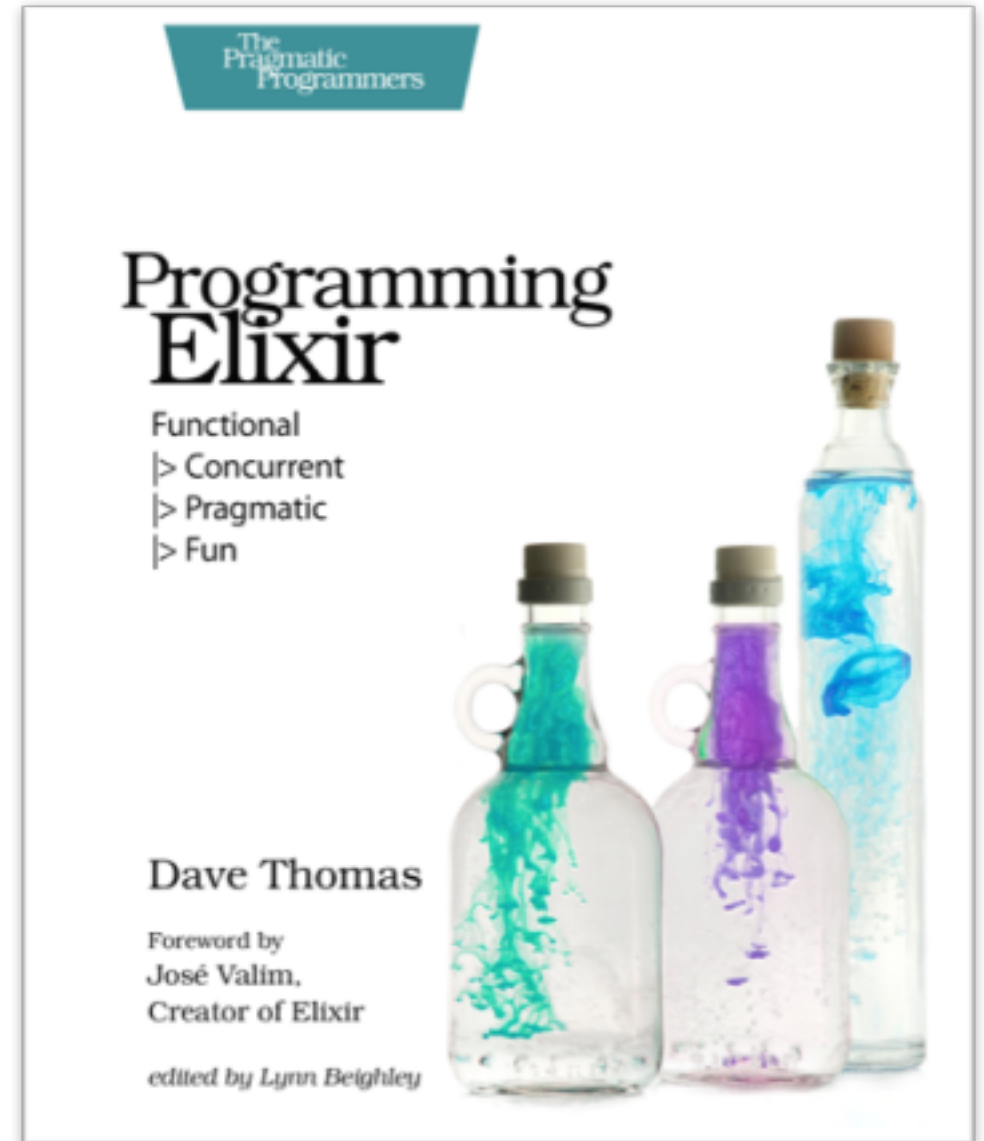
- Удобство разработки
- Простота изучения
- Постоянное развитие



Так появился Elixir...

Сила языка Erlang и
элегантность Ruby

- Создан в 2012 году
- Автор Jose Valim
- Бонусы:
 - Скорость, многоядерность
 - Метaprogramмирование
 - Инструменты mix, hex
 - Сообщество и документация
 - Все библиотеки Erlang



Install

Mac:

```
$ brew update
```

```
$ brew install elixir
```

Windows:

<https://repo.hex.pm/elixir-websetup.exe>

hello_world.ex

```
defmodule HelloWorld do
  def say_it do
    IO.puts "Hello World"
  end
end
```

```
$ iex
```

Interactive Elixir (1.3.2)

```
iex(1)> c("hello_world.ex")
```

```
[HelloWorld]
```

```
iex(2)> HelloWorld.say_it
```

```
Hello World
```

```
:ok
```

```
iex(3)>
```

ОСНОВНЫЕ ТИПЫ

Elixir поддерживает динамическое типизирование

123	# integer
0x1F	# integer
123.34	# float
true	# boolean
:atom	# atom
"elixir"	# string
[1, 2, 3]	# list
{1, 2, 3}	# tuple
%{a: 1, b: 2}	# map
~r/abc/	# sigil

Pattern matching

```
iex> x = 1
```

```
1
```

```
iex> 1 = x
```

```
1
```

```
iex> 2 = x
```

```
** (MatchError) no match of right hand side  
value: 1
```

Pattern matching

```
iex> {a, b, c} = {:hello, "world", 42}  
{:hello, "world", 42}
```

```
iex> a  
:hello
```

```
iex> b  
"world"
```

Pattern matching

```
iex> {:ok, result} = {:ok, 13}
```

```
  {:ok, 13}
```

```
iex> result
```

```
  13
```

```
iex> {:ok, result} = {:error, :oops}
```

```
  ** (MatchError) no match of right hand side  
  value: {:error, :oops}
```

Pattern matching

```
# flattens a list
def flatten([]), do: []
def flatten([head | tail]) when is_list(head), do: flatten(head) ++ flatten(tail)
def flatten([head | tail]), do: [head | flatten(tail)]
```


Factorial demo

List comprehensions

```
iex> for n <- [1, 2, 3, 4], do: n * n
```

```
[1, 4, 9, 16]
```

```
iex> values = [good: 1, good: 2, bad: 3, good: 4]
```

```
iex> for {:good, n} <- values, do: n * n
```

```
[1, 4, 16]
```

```
iex> for n <- 1..100, rem(n, 3) == 0, do: n
```

```
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39,  
42, 45, 48, 51, 54, 57, 60, 63, 66, 69, 72, 75, 78,  
81, 84, 87, 90, 93, 96, 99]
```

List comprehensions

```
defmodule Comprehensions do
  def list(x, y) do
    for n <- x..y, do: n
  end
  def primes(n) do
    l = list(2, n)
    for x <- l, _is_prime(x), do: x
  end
  defp _is_prime(x) do
    check = for y <- 2..x, rem(x, y) == 0, do: y
    length(check) == 1
  end
end
```

Default arguments

```
defmodule Concat do
  def join(a, b, sep \\ " ") do
    a <> sep <> b
  end
end
```

```
I0.puts Concat.join("Hello", "world") #=> Hello world
I0.puts Concat.join("Hello", "world", "_") #=> Hello_world
```

Function capturing

```
iex> fun = &(&1 + 1)
```

```
#Function<6.71889879/1 in :erl_eval.expr/5>
```

```
iex> fun.(1)
```

```
2
```

Pipe operator

```
defmodule Crunch do
  def of(something) do
    something
    |> filter
    |> sort
    |> group
    |> count
  end
end
```

Mix

```
$ mix new project
```

```
* creating README.md
```

```
* creating .gitignore
```

```
* creating mix.exs
```

```
* creating config
```

```
* creating config/config.exs
```

```
* creating lib
```

```
* creating lib/project.ex
```

```
* creating test
```

```
* creating test/test_helper.exs
```

```
* creating test/project_test.exs
```

Your Mix project was created successfully.

You can use "mix" to compile it, test it, and more:

```
cd project
```

```
mix test
```

Run "mix help" for more commands.

Processes

```
iex> send self(), {:hello, "world"}  
      {:hello, "world"}  
iex> receive do  
...> {:hello, msg} -> msg  
...> {:world, msg} -> "won't match"  
...> end  
      "world"
```


Processes

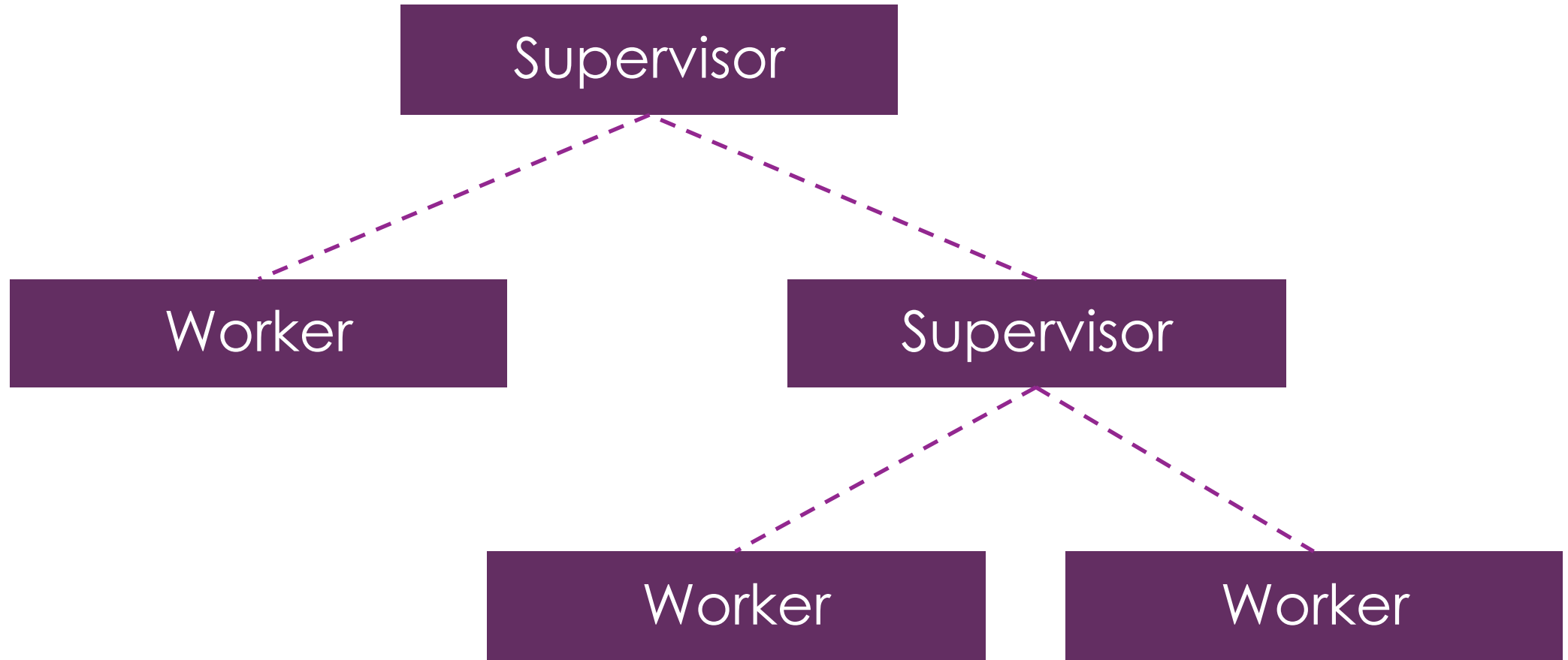
`spawn`(Module, :function, [params])

`spawn_link`(Module, :function, [params])

`spawn_monitor`(Module, :function, [params])

SpawnOne demo

OTP – Open Telecom Platform



Документация

The screenshot shows the Elixir v1.3.2 documentation website. On the left is a dark sidebar with the Elixir logo and a search bar. Below the search bar is a list of navigation links: PAGES, MODULES (highlighted with a purple bar), EXCEPTIONS, PROTOCOLS, inspect, Integer, and Kernel. Under Kernel, there are links for Summary, Functions, and Macros. Further down are links to Kernel.ParallelCompiler, Kernel.ParallelRequire, Kernel.SpecialForms, Keyword, List, Macro, Macro.Env, Map, MapSet, Module, NaiveDateTime, Node, OptionParser, Path, and Port.

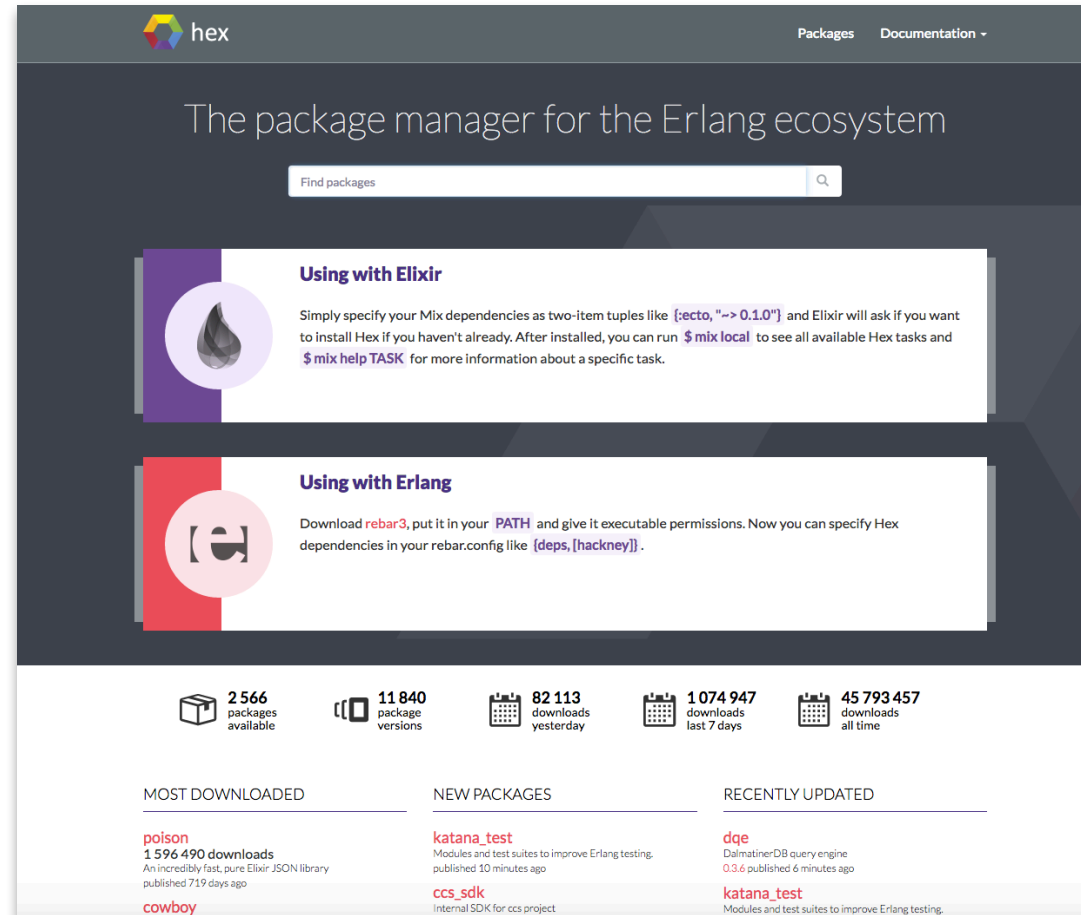
The main content area displays the documentation for the `is_atom(term)` function. At the top, the function signature `is_atom(term)` is shown in a light gray box with a code icon on the right. Below this, the section **Specs** contains the type signature `is_atom(term) :: boolean` in a light gray box. The text explains that the function returns `true` if `term` is an atom, otherwise it returns `false`. It also notes that the function is allowed in guard tests and is inlined by the compiler.

Below the `is_atom` section, the documentation for the `is_binary(term)` function begins. Its signature `is_binary(term)` is shown in a light gray box. The **Specs** section shows the type signature `is_binary(term) :: boolean`. The text explains that it returns `true` if `term` is a binary, otherwise it returns `false`. It also states that a binary always contains a complete number of bytes and that the function is allowed in guard tests and is inlined by the compiler.

The **Examples** section shows a code block with the following Elixir REPL interactions:

```
iex> is_binary "foo"  
true  
iex> is_binary <<1::3>>  
false
```

Packages and libraries



The screenshot shows the Hex website, which is the package manager for the Erlang ecosystem. The header includes the Hex logo and navigation links for 'Packages' and 'Documentation'. The main heading is 'The package manager for the Erlang ecosystem', followed by a search bar labeled 'Find packages'. Below this, there are two sections: 'Using with Elixir' and 'Using with Erlang'. The 'Using with Elixir' section explains how to specify Mix dependencies as two-item tuples like `{:ecto, "~> 0.1.0"}` and mentions running `$ mix local` to see available tasks and `$ mix help TASK` for more information. The 'Using with Erlang' section explains how to download `rebar3`, put it in your `PATH`, and give it executable permissions, then specify Hex dependencies in your `rebar.config` like `[deps, [hackney]]`. At the bottom, there are statistics: 2566 packages available, 11840 package versions, 82113 downloads yesterday, 1074947 downloads last 7 days, and 45793457 downloads all time. Below these are three columns: 'MOST DOWNLOADED' featuring `poison` (1596490 downloads), `cowboy`, and `katana_test`; 'NEW PACKAGES' featuring `katana_test` and `ccs_sdk`; and 'RECENTLY UPDATED' featuring `dqe` and `katana_test`.

hex Packages Documentation

The package manager for the Erlang ecosystem

Find packages

Using with Elixir

Simply specify your Mix dependencies as two-item tuples like `{:ecto, "~> 0.1.0"}` and Elixir will ask if you want to install Hex if you haven't already. After installed, you can run `$ mix local` to see all available Hex tasks and `$ mix help TASK` for more information about a specific task.

Using with Erlang

Download `rebar3`, put it in your `PATH` and give it executable permissions. Now you can specify Hex dependencies in your `rebar.config` like `[deps, [hackney]]`.

2566 packages available 11840 package versions 82113 downloads yesterday 1074947 downloads last 7 days 45793457 downloads all time

MOST DOWNLOADED

poison
1596490 downloads
An incredibly fast, pure Elixir JSON library
published 719 days ago

cowboy

NEW PACKAGES

katana_test
Modules and test suites to improve Erlang testing.
published 10 minutes ago

ccs_sdk
Internal SDK for ccs project.

RECENTLY UPDATED

dqe
Dalmatiner-DB query engine
0.3.6 published 6 minutes ago

katana_test
Modules and test suites to improve Erlang testing.



Similar to Ruby on Rails and different

- MVC
- Plugins
- Migrations
- Eex ~= Erb
- Path helpers
- Router
- Schema
- Generators
- No Active Record
- Sockets
- Channels
- Views
- 10-40x faster

\$ mix phoenix.new awesome

Thank you