325
50
575
300
20
25
30
35
PLEASE STAND BY

Catalyze Change

José Valim @josevalim

Dave Thomas @pragdave

We Have A Problem

- We are sitting on the solution to many of today's problems
- We've had these solutions for 15 years
- Yet we are still marginal

Why?

Dear Abbey…

I keep trying to apply Elixir/Erlang to problems I have, and I keep running up against issues with the terrible state of the ecosystem. Whatever the Erlang devs have been up to for the last 25 years, it certainly wasn't focusing on being a citizen of the modern internet.

I've killed many, many hours deciphering bad documentation and fighting bugs in XML and HTTP libraries over the past several weeks. Most recently I thought I had an idea which would be a perfect application of Elixir/Erlang, and then discovered that its sole IMAP client library is moribund and considered unusable by the one person I could find who has tried it.

Now, obviously, "the libraries suck" is a bad argument against a language - Ruby was in just as bad a state back in 2001 when I came to it. But there were obsession with arity. And all of them have Actor/CSP/ Pi Calculus concurrency abstractions baked-in, just like Erlang.

Given all this, it seems like the sole reason to invest time into Elixir over some of these other languages is on the faith that the VM is simply that much better than the JVM, or the Go runtime.

At this point, my intuition is screaming at me to ditch the Erlang VM and focus my concurrent language energy on Clojure and/or Go. I keep worrying that I'll fight my way through the ecosystem issues, only to find myself with a bunch of code that runs on a VM that isn't materially better than the JVM, and can't be instrumented and tuned the way the JVM can.

Not a Random Rant

- We hear it a lot
- From well-known, knowledgeable people

We Need to Reconcile

we are sitting on the solution to many of today's problems

vs.

"ditch the Erlang VM and focus my concurrent language energy on Clojure and/or Go”

Why Do We Care?

- We're happy
  - We have what we need
  - It works for us
- Someone else's problem

But We Should Care

- For the good of ourselves
- For the long-term survival of our culture.
- More conferences, more companies, more jobs, more clever ideas…
- For the good of world!

We’re Better

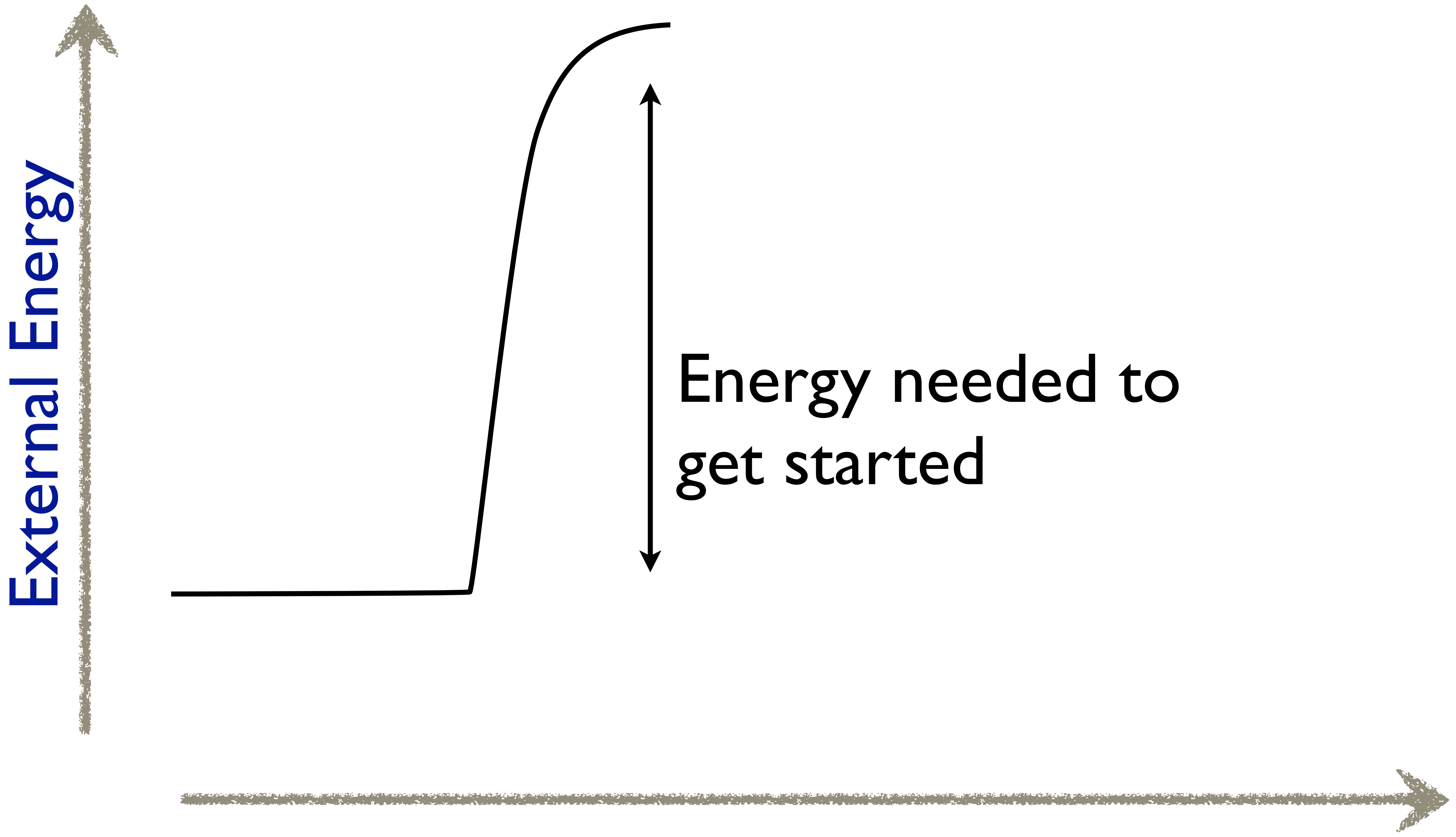
- We have the best VM and the best language
- So we don't have to worry about the surface features.

But the barriers to entry are too high

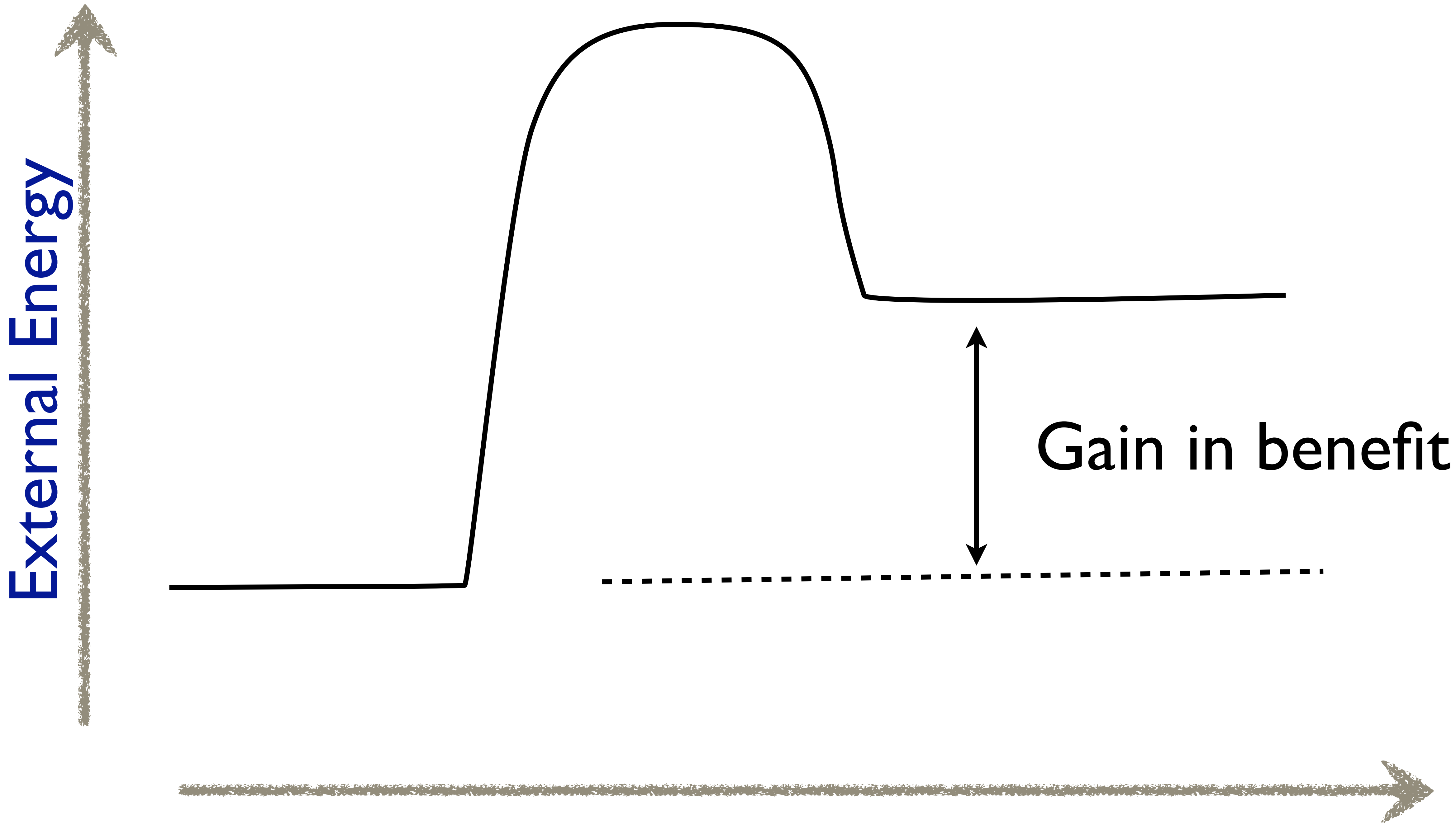
Activation Energy

External Energy

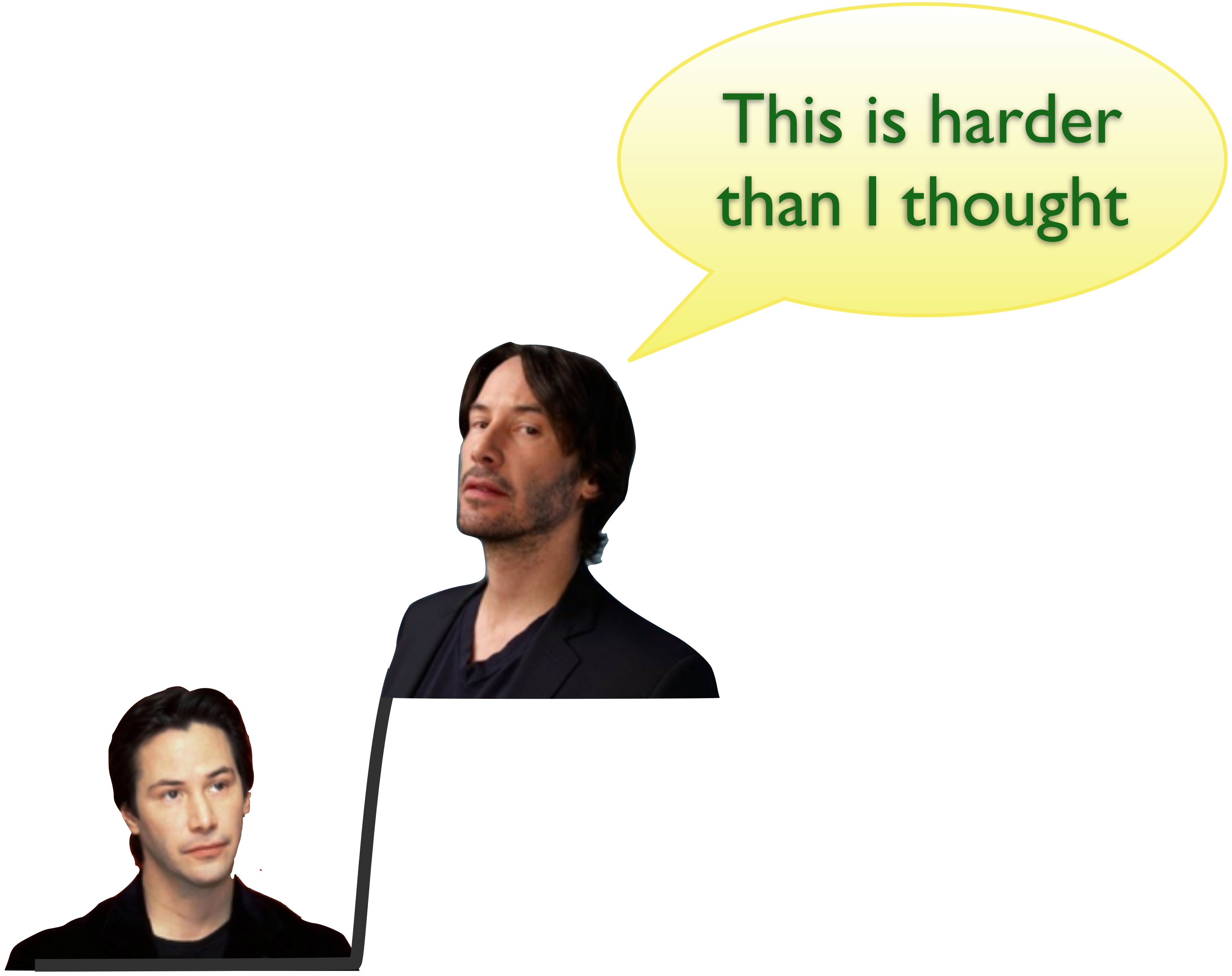
Activation Energy



Activation Energy

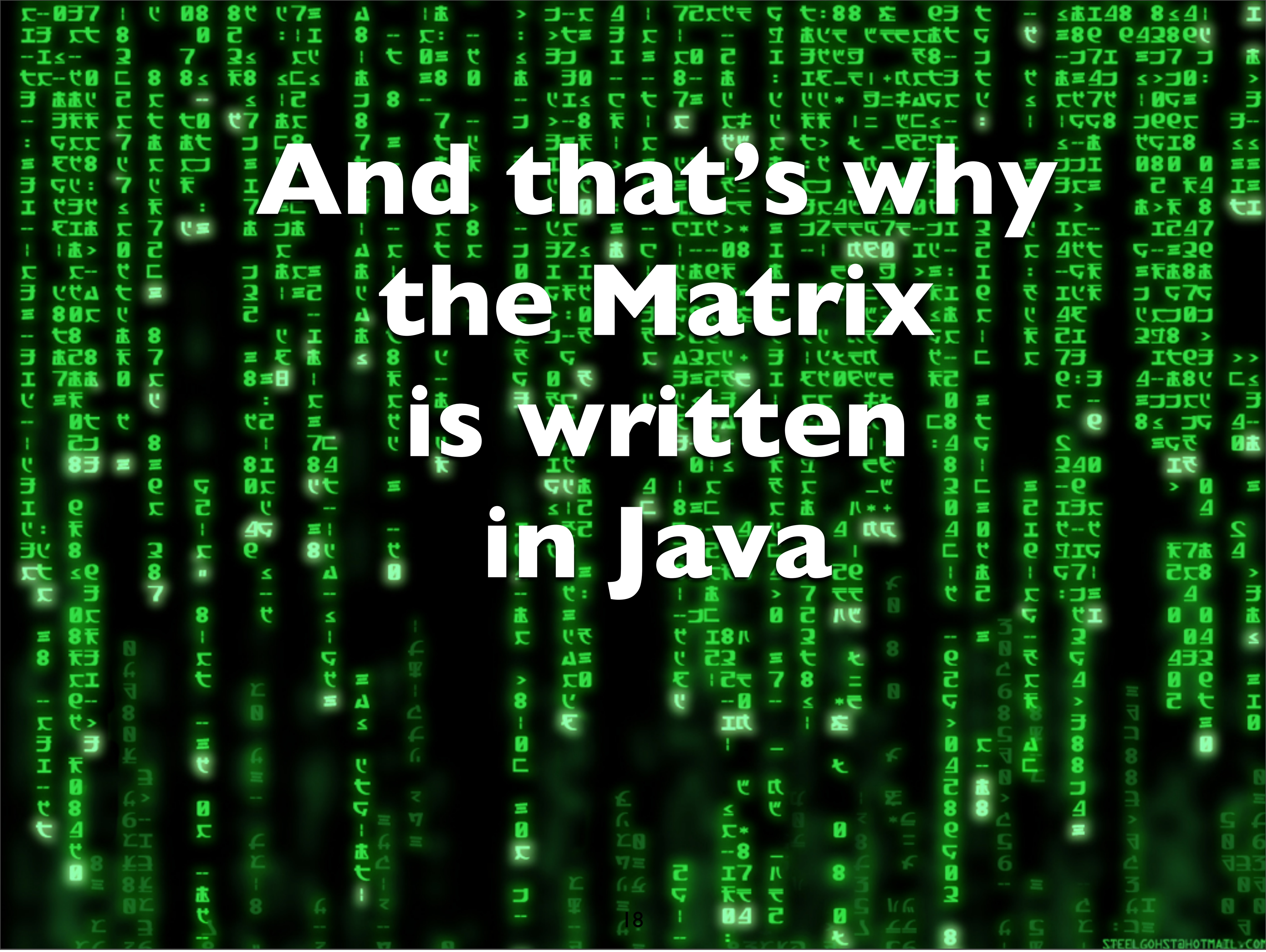


Hmmm…
Erlang looks
interesting


This is harder than I thought

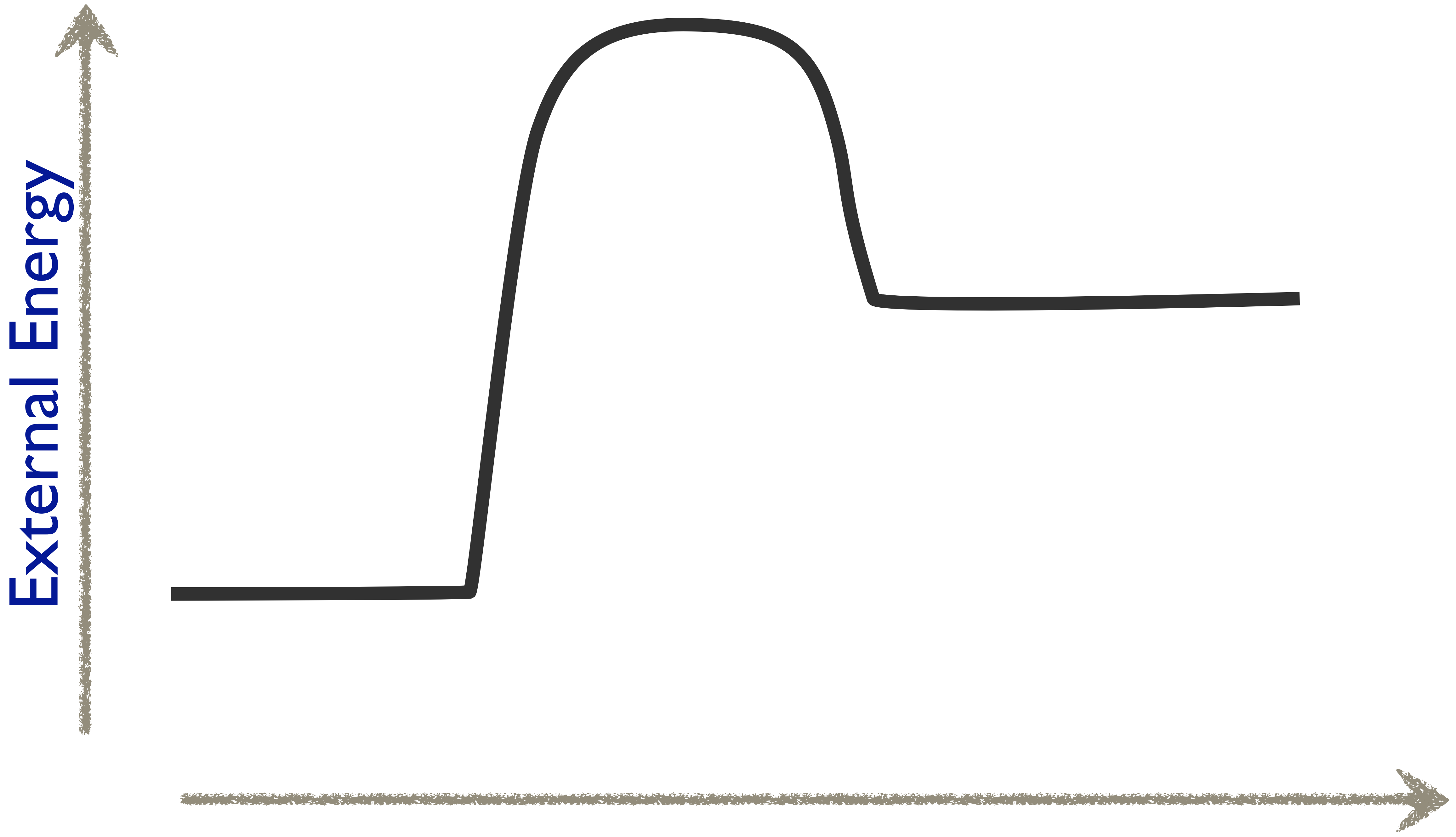
I hate feeling stupid

http://halfagiraffe.tv/wp-content/uploads/2010/10/KeanuReeves1.jpg
http://us.cdn200.fansshare.com/photo/keanureeves/keanu-reeves-83926830.jpg
http://wallpaperhdfree.com/wp-content/uploads/2013/03/Angry-Keanu-Reeves.jpg
http://images2.fanpop.com/image/photos/9200000/Keanu-Reeves-keanu-reeves-9231566-1280-1024.jpg
http://wallpaperhdfree.com/wp-content/uploads/2013/03/Keanu-Reeves-Best-2013-Wallpaper.jpg
http://us.cdn281.fansshare.com/photos/keanureeves/keanu-reeves-wallpaper-1469967791.jpg

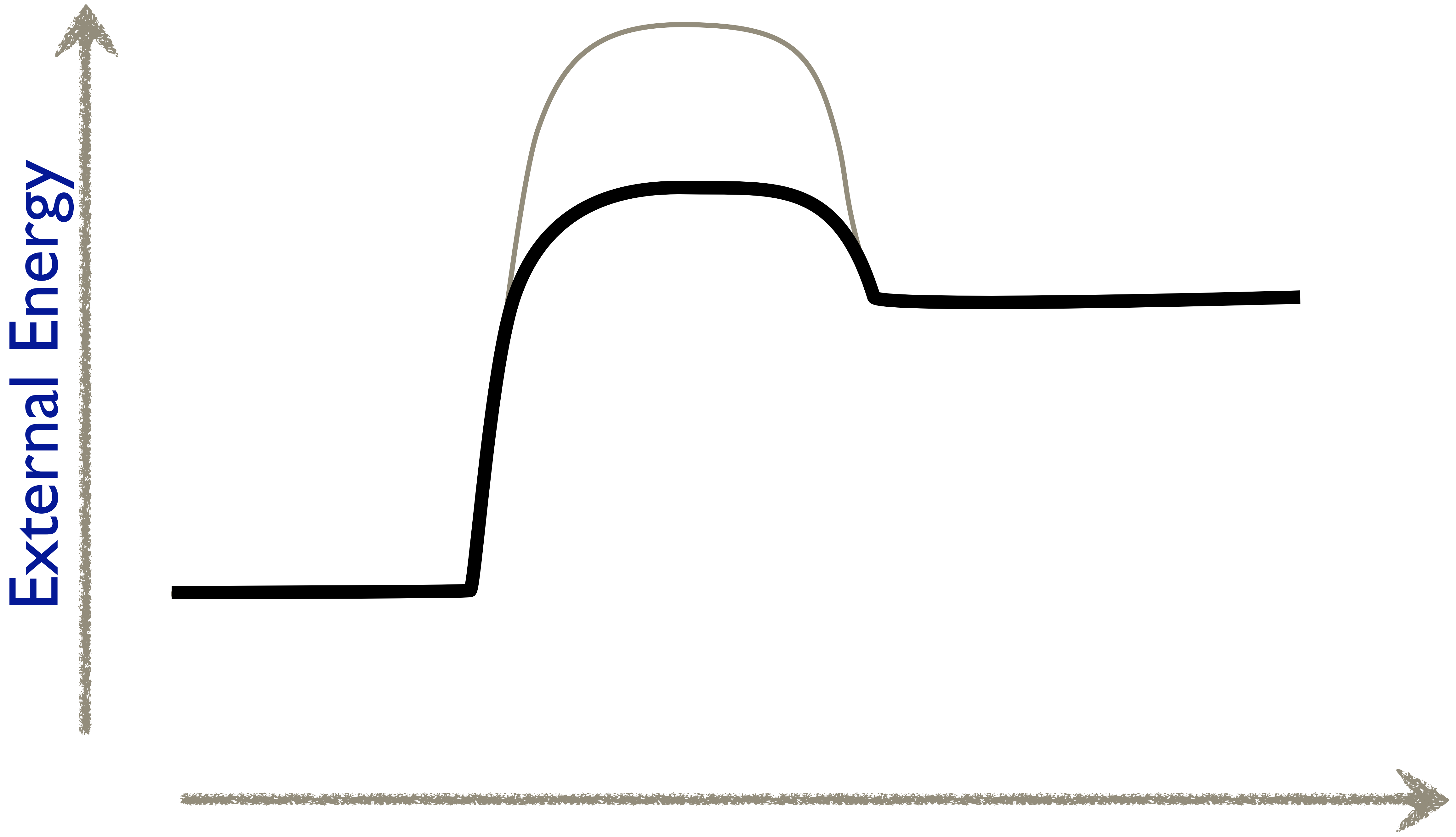


And that's why the Matrix is written in Java

Lower the Barrier

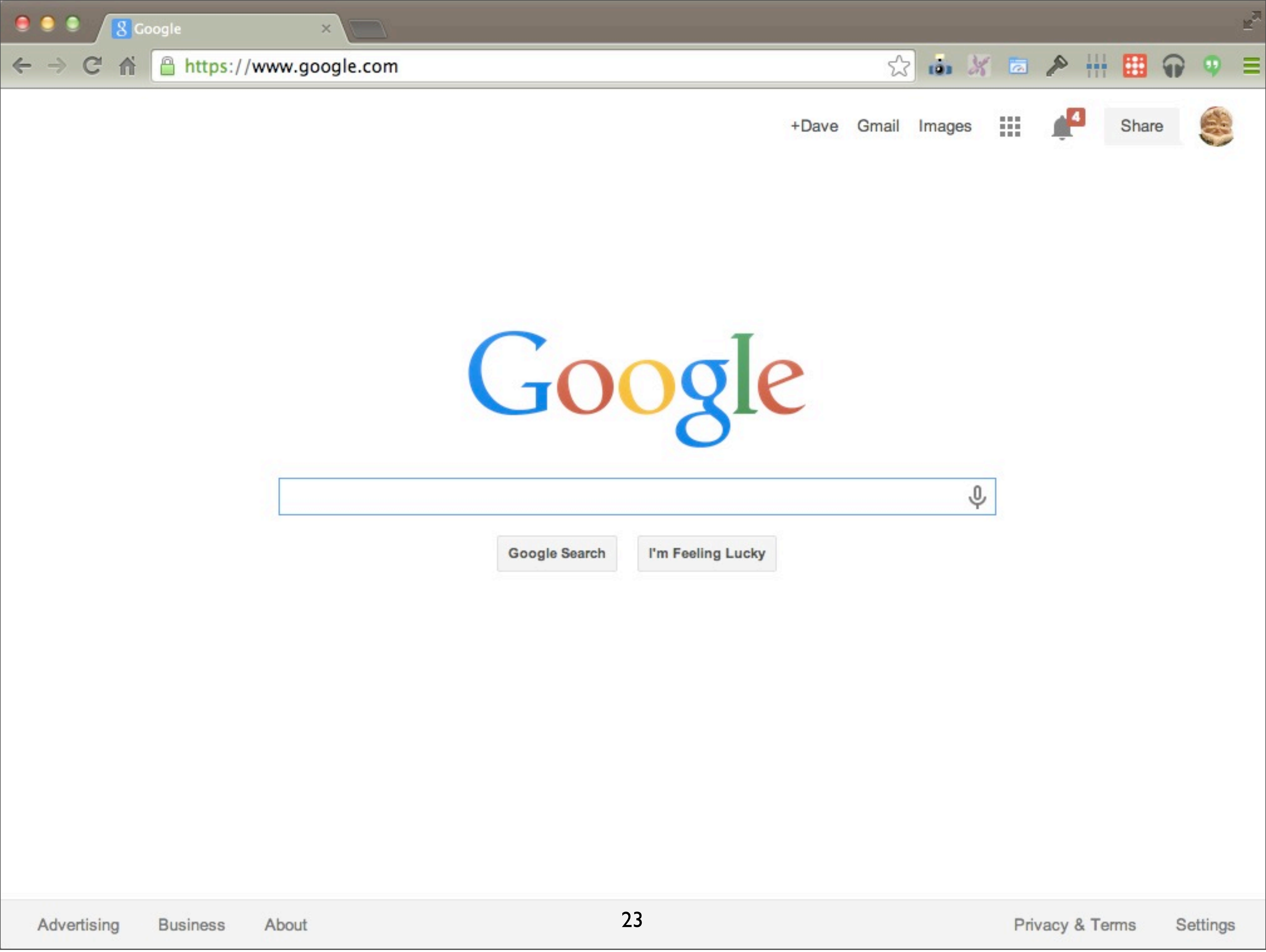


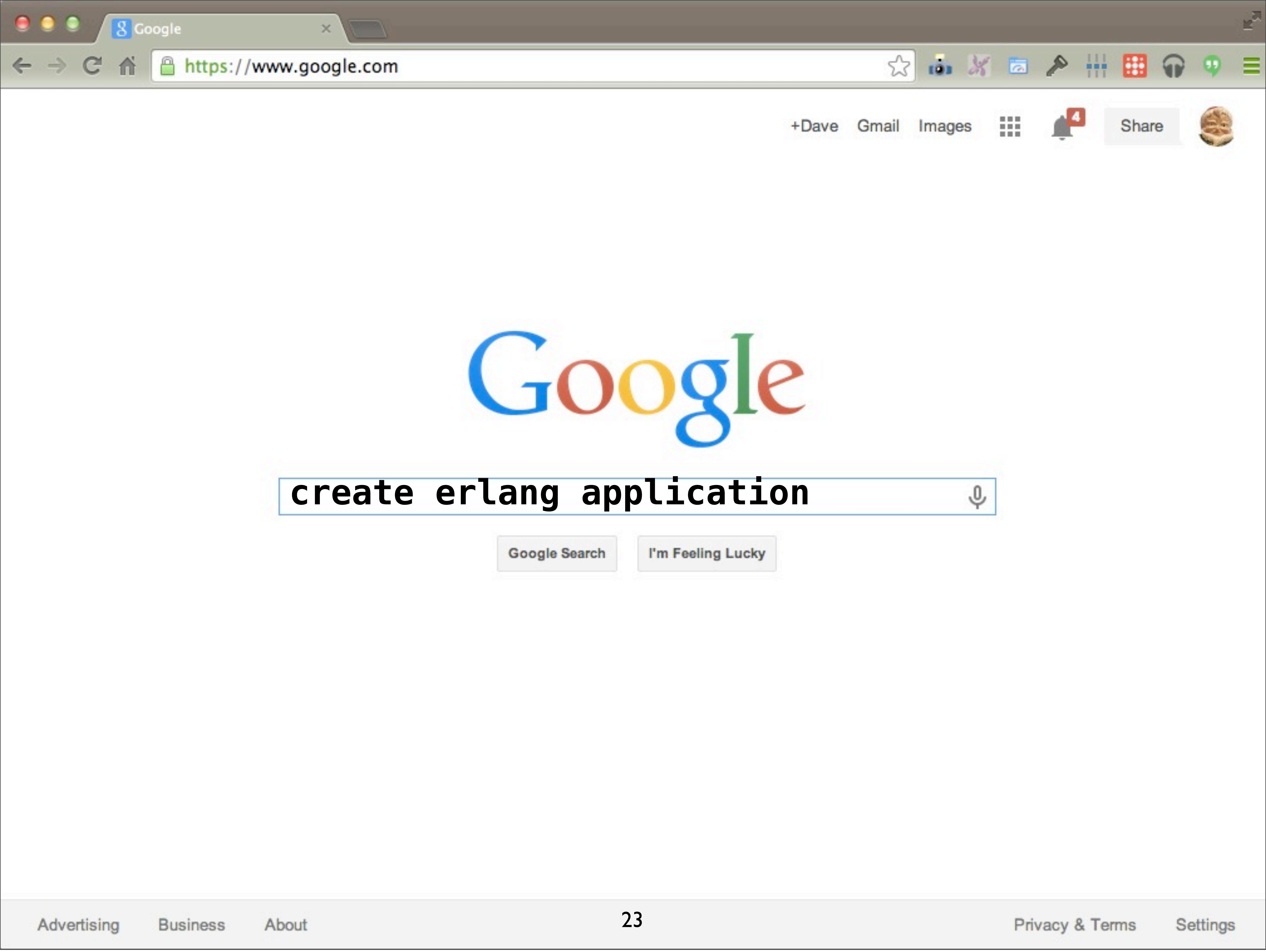
Catalyst Reduces Input



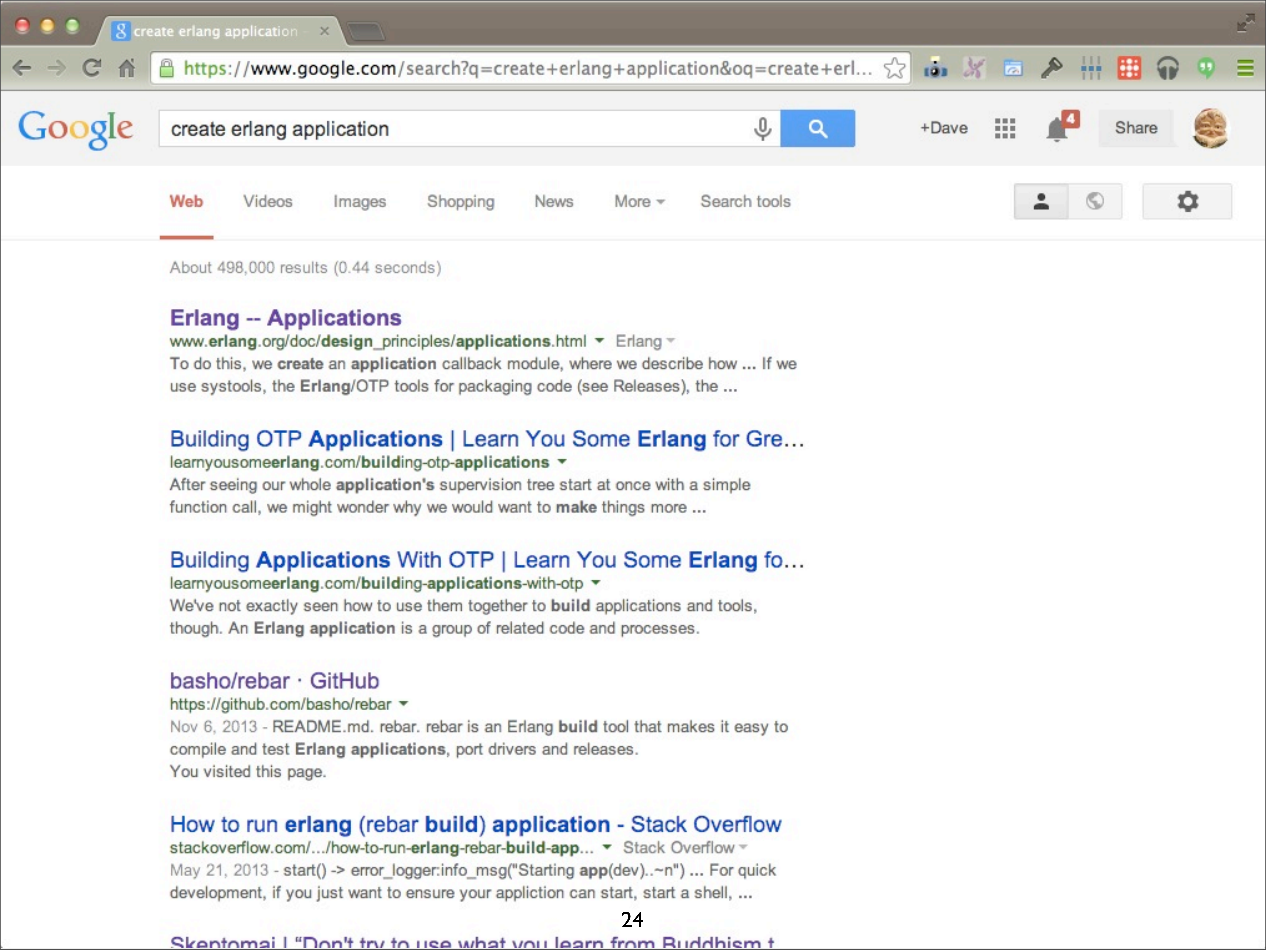
You Are The Catalyst

So Let's Put On our Outsider Hats


Google
https://www.google.com
+Dave
Gmail
Images
4
Share
Google
Google Search
I'm Feeling Lucky
Advertising
Business
About
Privacy & Terms
Settings



create erlang application



create erlang application

https://www.google.com/search?q=create+erlang+application&oq=create+erl...

create erlang application

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html ▾ Erlang ▾
To do this, we create an application callback module, where we describe how ... If we use systools, the Erlang/OTP tools for packaging code (see Releases), the ...

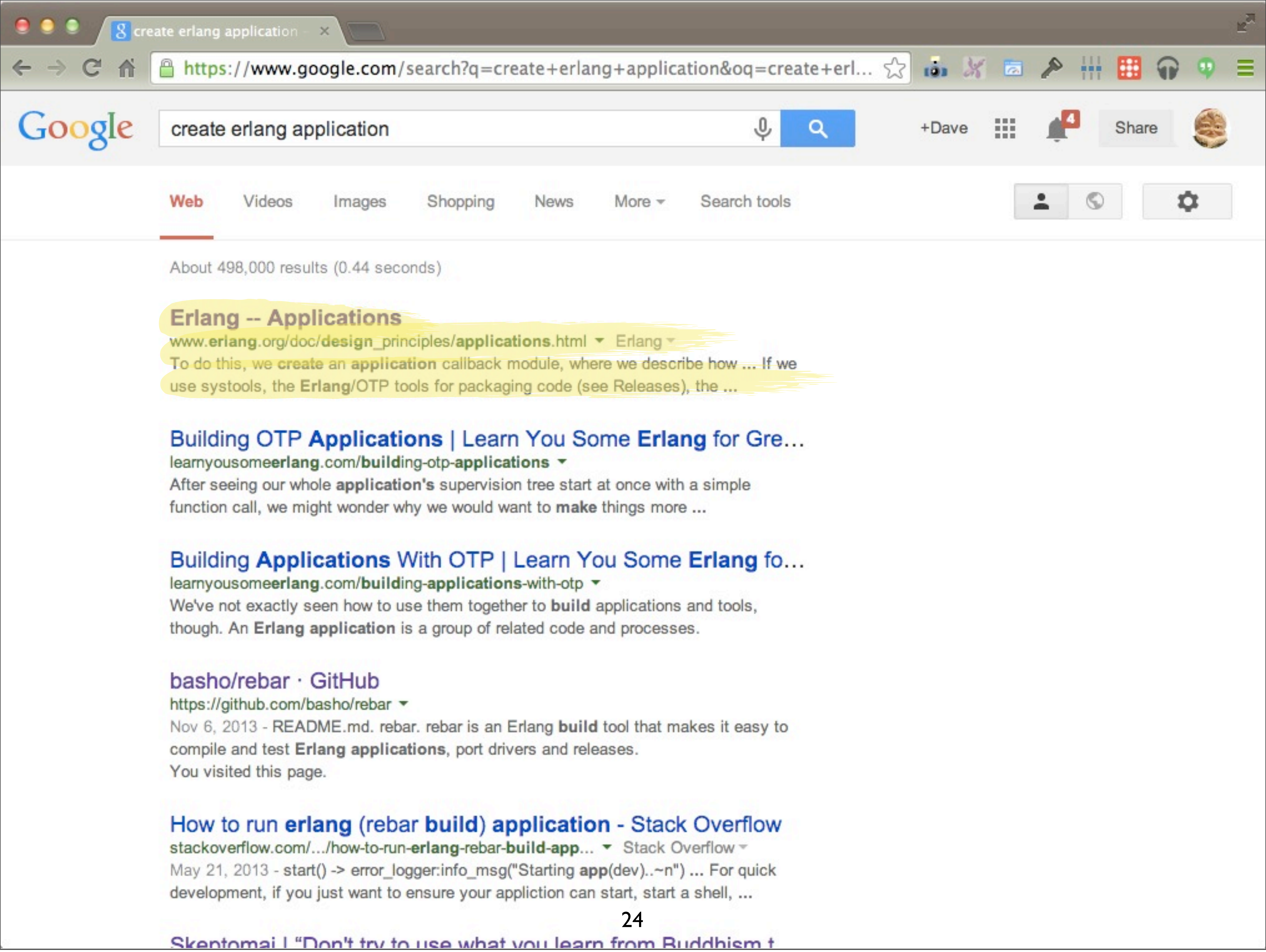
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications ▾
After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp ▾
We've not exactly seen how to use them together to build applications and tools, though. An Erlang application is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar ▾
Nov 6, 2013 - README.md. rebar. rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... ▾ Stack Overflow ▾
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t


create erlang application
https://www.google.com/search?q=create+erlang+application&oq=create+erl...
Google
create erlang application
+Dave
4
Share
Web
Videos
Images
Shopping
News
More
Search tools

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html ▾ Erlang ▾
To do this, we create an application callback module, where we describe how ... If we use systools, the Erlang/OTP tools for packaging code (see Releases), the ...

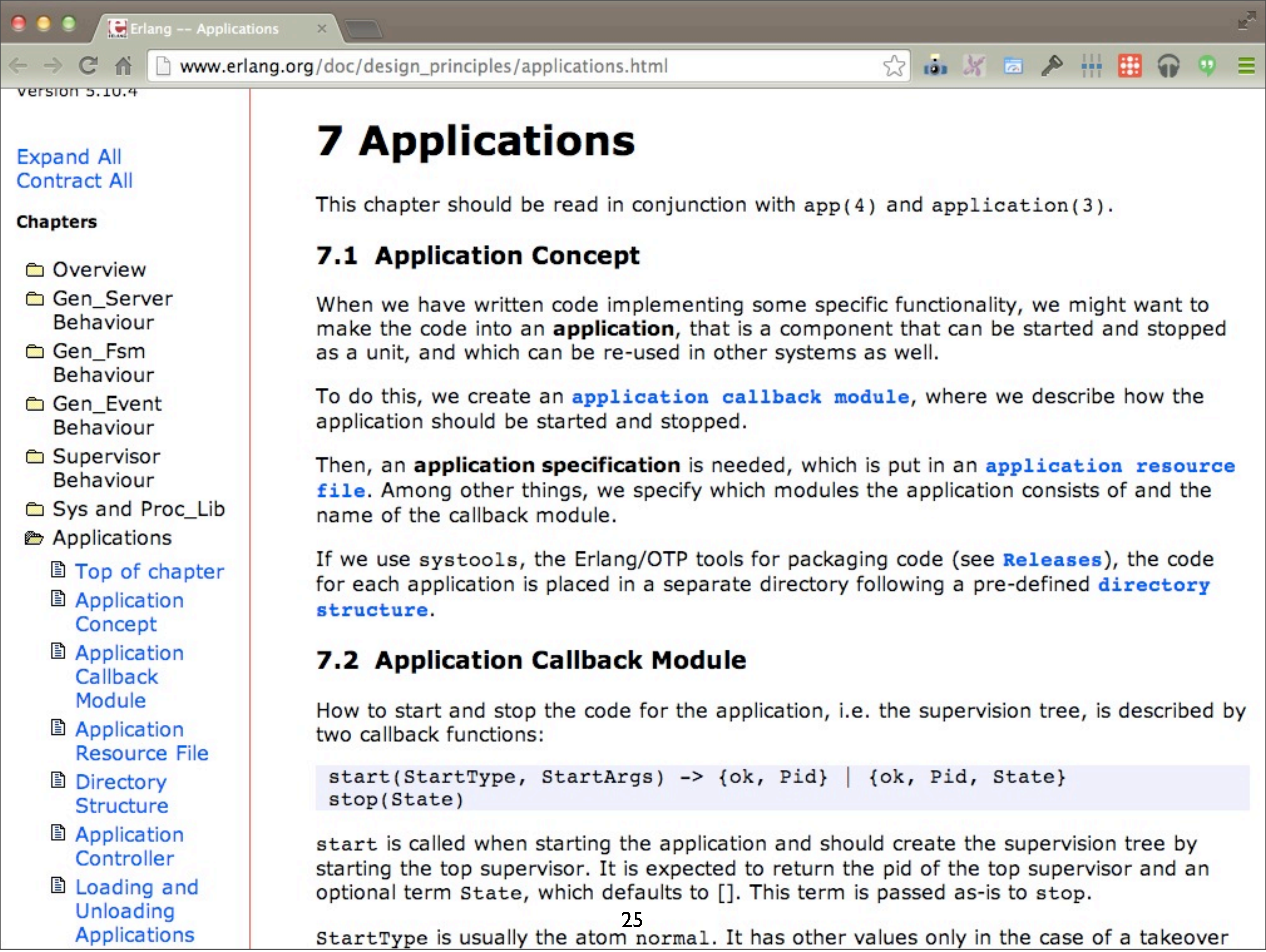
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications ▾
After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp ▾
We've not exactly seen how to use them together to build applications and tools, though. An Erlang application is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar ▾
Nov 6, 2013 - README.md. rebar. rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... ▾ Stack Overflow ▾
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t



Chapters

7 Applications

This chapter should be read in conjunction with `app(4)` and `application(3)`.

7.1 Application Concept

When we have written code implementing some specific functionality, we might want to make the code into an **application**, that is a component that can be started and stopped as a unit, and which can be re-used in other systems as well.

To do this, we create an **application callback module**, where we describe how the application should be started and stopped.

Then, an **application specification** is needed, which is put in an **application resource file**. Among other things, we specify which modules the application consists of and the name of the callback module.

If we use `systools`, the Erlang/OTP tools for packaging code (see **Releases**), the code for each application is placed in a separate directory following a pre-defined **directory structure**.

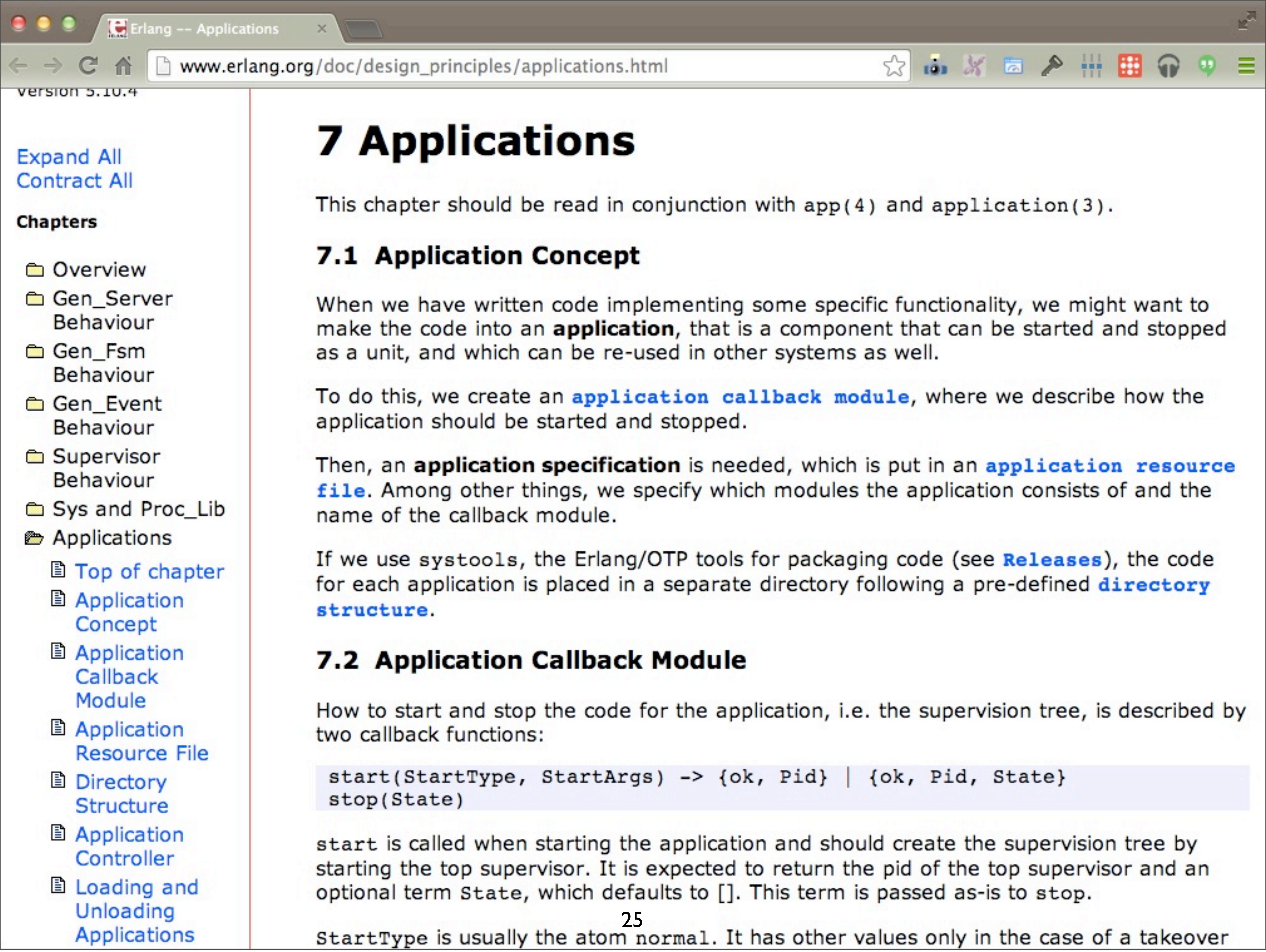
7.2 Application Callback Module

How to start and stop the code for the application, i.e. the supervision tree, is described by two callback functions:

```
start(StartType, StartArgs) -> {ok, Pid} | {ok, Pid, State}
stop(State)
```

`start` is called when starting the application and should create the supervision tree by starting the top supervisor. It is expected to return the pid of the top supervisor and an optional term `State`, which defaults to []. This term is passed as-is to `stop`.

`StartType` is usually the atom `normal`. It has other values only in the case of a takeover



Erlang -- Applications

www.erlang.org/doc/design_principles/applications.html

Expand All
Contract All

Chapters

7 Applications

This chapter should be read in conjunction with `app(4)` and `application(3)`.

7.1 Application Concept

When we have written code implementing some specific functionality, we might want to make the code into an **application**, that is a component that can be started and stopped as a unit, and which can be re-used in other systems as well.

To do this, we create an **application callback module**, where we describe how the application should be started and stopped.

Then, an **application specification** is needed, which is put in an **application resource file**. Among other things, we specify which modules the application consists of and the name of the callback module.

If we use `systools`, the Erlang/OTP tools for packaging code (see **Releases**), the code for each application is placed in a separate directory following a pre-defined **directory structure**.

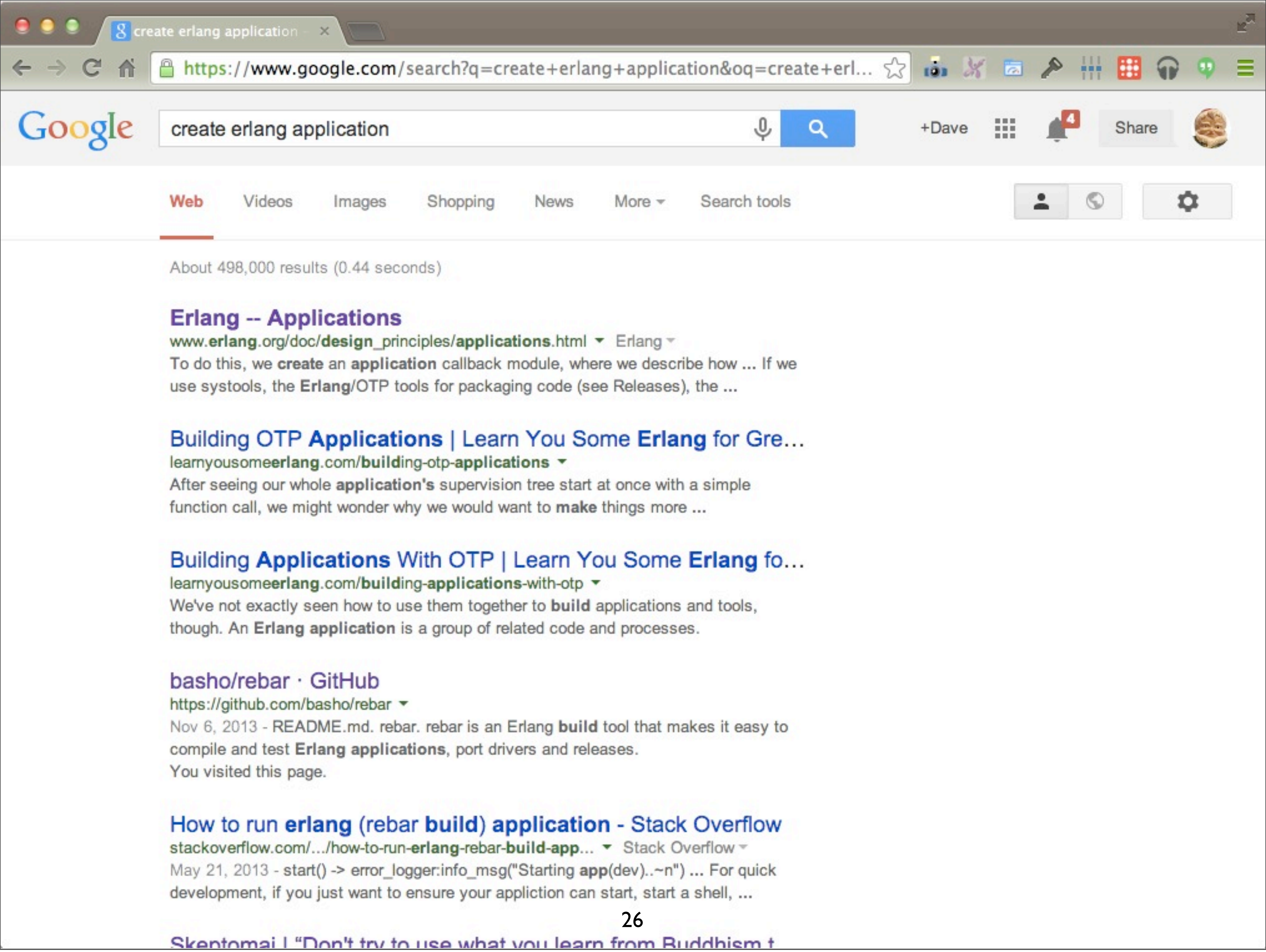
7.2 Application Callback Module

How to start and stop the code for the application, i.e. the supervision tree, is described by two callback functions:

```
start(StartType, StartArgs) -> {ok, Pid} | {ok, Pid, State}
stop(State)
```

`start` is called when starting the application and should create the supervision tree by starting the top supervisor. It is expected to return the pid of the top supervisor and an optional term `State`, which defaults to []. This term is passed as-is to `stop`.

`StartType` is usually the atom `normal`. It has other values only in the case of a takeover


create erlang application
https://www.google.com/search?q=create+erlang+application&oq=create+erl...
create erlang application
+Dave
Share
Web
Videos
Images
Shopping
News
More
Search tools

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html Erlang
To do this, we create an application callback module, where we describe how ... If we use systools, the Erlang/OTP tools for packaging code (see Releases), the ...

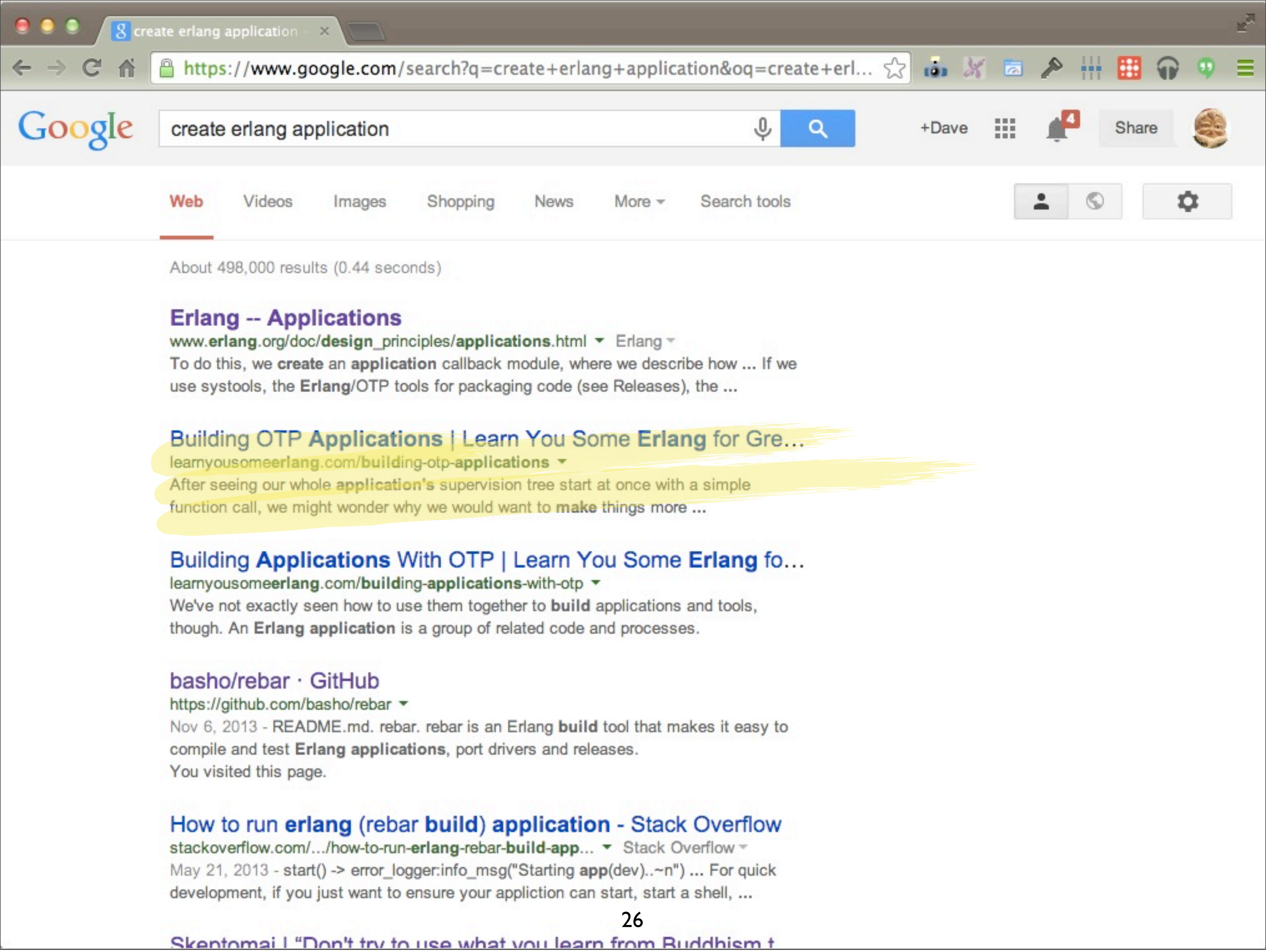
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications
After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp
We've not exactly seen how to use them together to build applications and tools, though. An Erlang application is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar
Nov 6, 2013 - README.md. rebar. rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... Stack Overflow
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t


create erlang application
https://www.google.com/search?q=create+erlang+application&oq=create+erl...
Google
create erlang application
+Dave
4
Share
Web
Videos
Images
Shopping
News
More
Search tools

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html ▾ Erlang ▾
To do this, we create an application callback module, where we describe how ... If we use systools, the Erlang/OTP tools for packaging code (see Releases), the ...

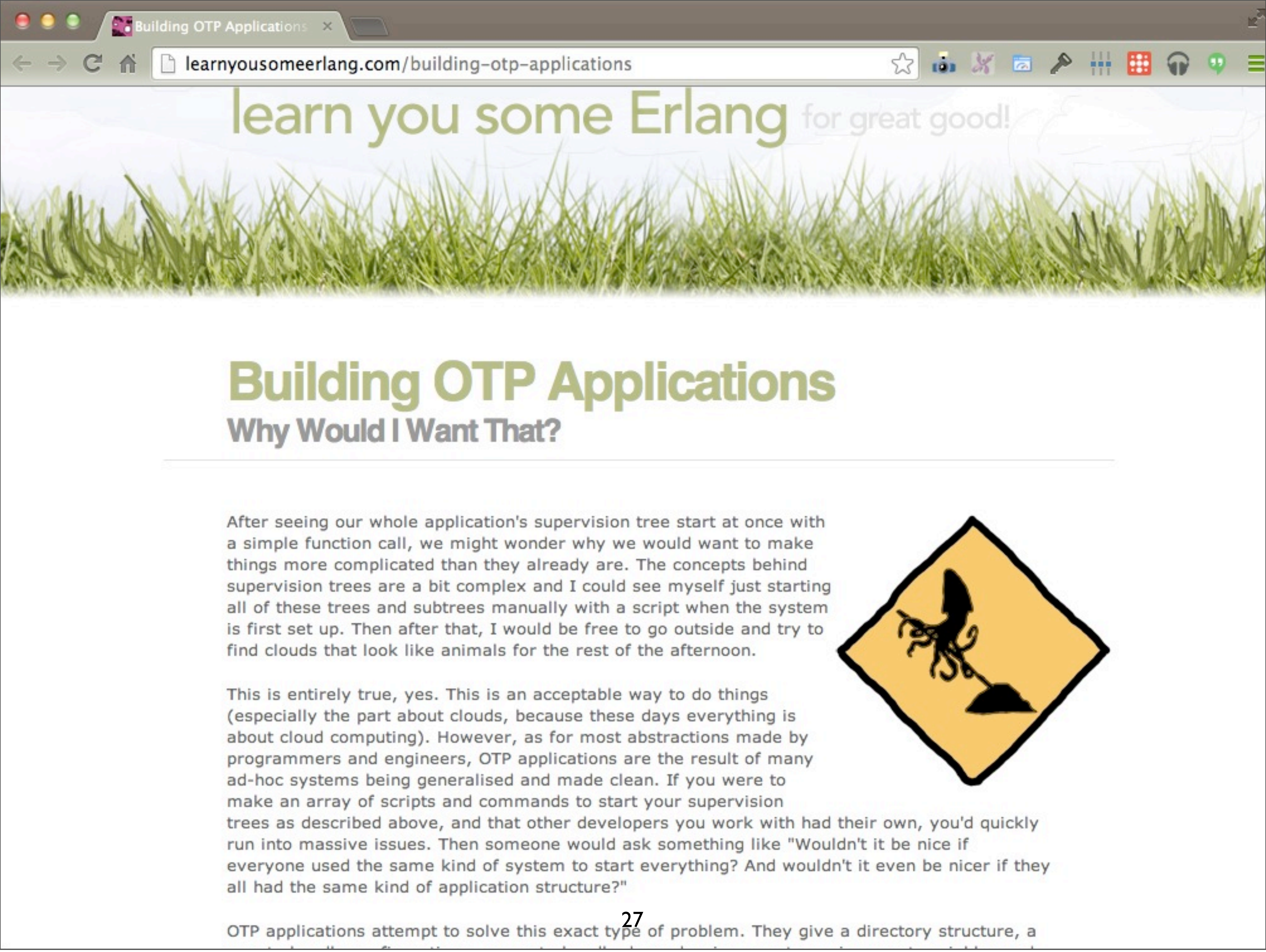
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications ▾
After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp ▾
We've not exactly seen how to use them together to build applications and tools, though. An Erlang application is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar ▾
Nov 6, 2013 - README.md. rebar. rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... ▾ Stack Overflow ▾
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t


Building OTP Applications
learnyousomeerlang.com/building-otp-applications

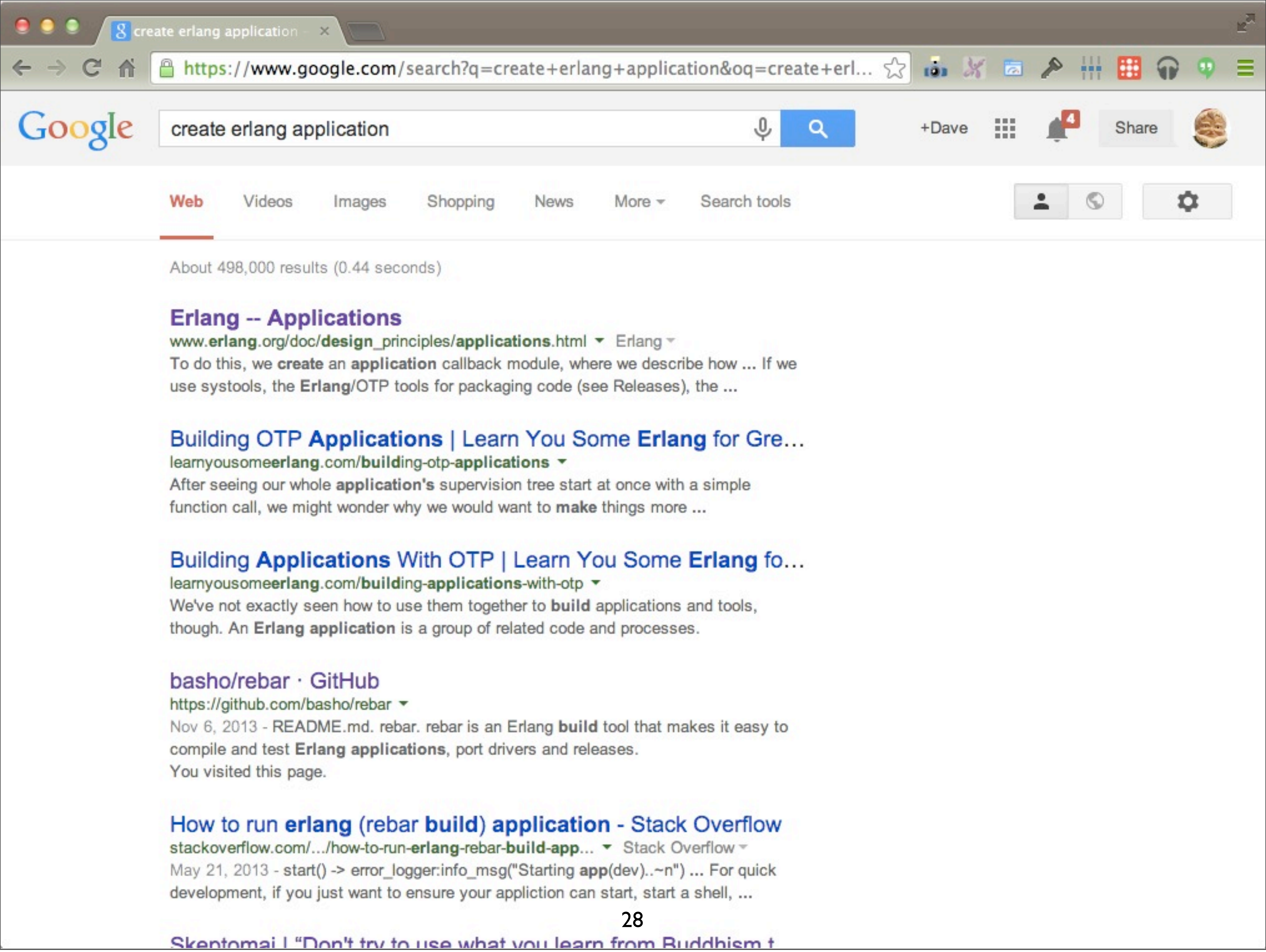
learn you some Erlang for great good!

Building OTP Applications
Why Would I Want That?

After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more complicated than they already are. The concepts behind supervision trees are a bit complex and I could see myself just starting all of these trees and subtrees manually with a script when the system is first set up. Then after that, I would be free to go outside and try to find clouds that look like animals for the rest of the afternoon.

This is entirely true, yes. This is an acceptable way to do things (especially the part about clouds, because these days everything is about cloud computing). However, as for most abstractions made by programmers and engineers, OTP applications are the result of many ad-hoc systems being generalised and made clean. If you were to make an array of scripts and commands to start your supervision trees as described above, and that other developers you work with had their own, you'd quickly run into massive issues. Then someone would ask something like "Wouldn't it be nice if everyone used the same kind of system to start everything? And wouldn't it even be nicer if they all had the same kind of application structure?"

OTP applications attempt to solve this exact type of problem. They give a directory structure, a


create erlang application
https://www.google.com/search?q=create+erlang+application&oq=create+erl...
Google
create erlang application
+Dave
Share
Web
Videos
Images
Shopping
News
More
Search tools

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html Erlang
To do this, we create an application callback module, where we describe how ... If we use systools, the Erlang/OTP tools for packaging code (see Releases), the ...

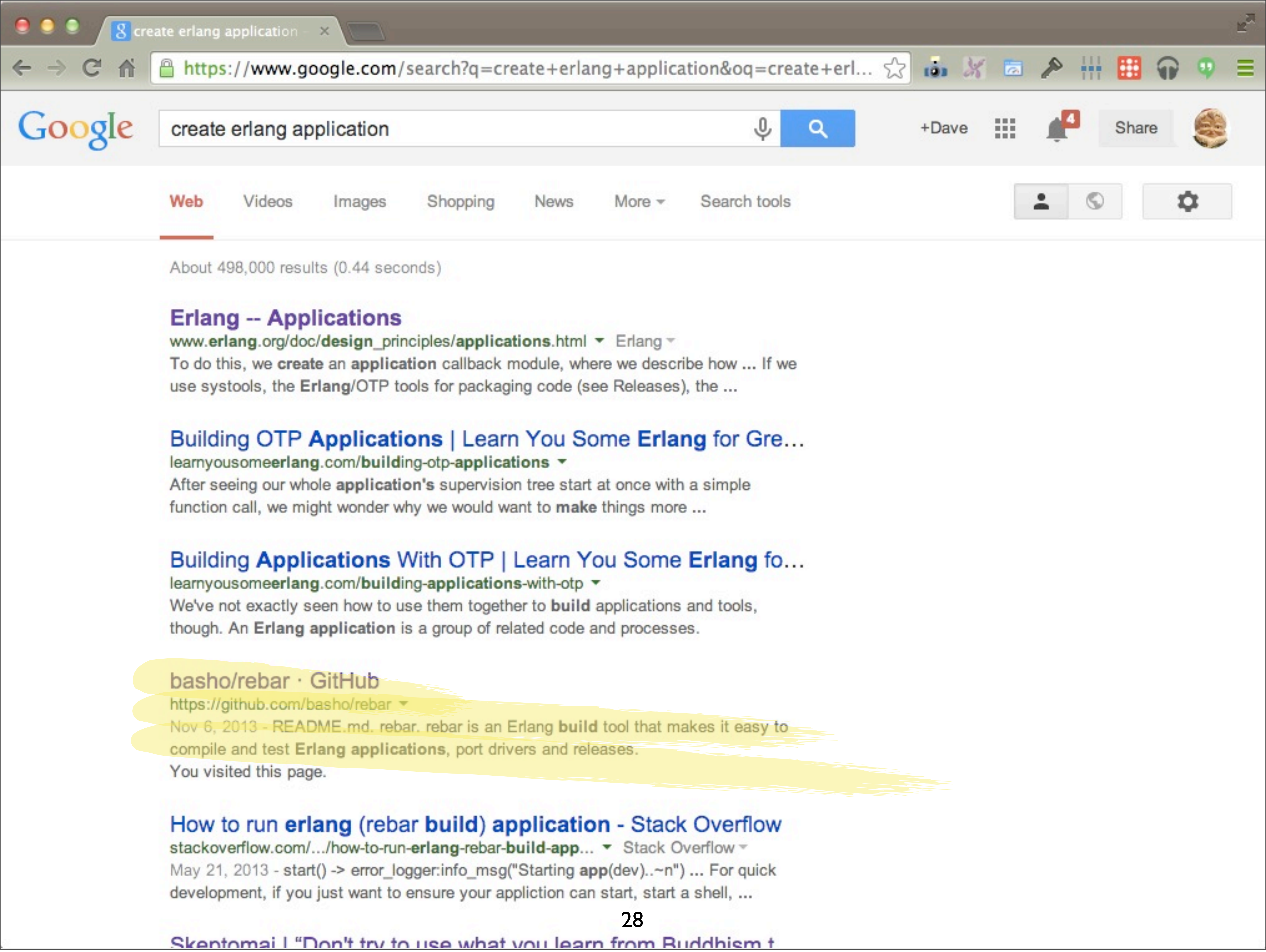
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications
After seeing our whole application's supervision tree start at once with a simple function call, we might wonder why we would want to make things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp
We've not exactly seen how to use them together to build applications and tools, though. An Erlang application is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar
Nov 6, 2013 - README.md. rebar. rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... Stack Overflow
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t



create erlang application

https://www.google.com/search?q=create+erlang+application&oq=create+erl...

create erlang application

+Dave · Share

Web · Videos · Images · Shopping · News · More · Search tools

About 498,000 results (0.44 seconds)

Erlang -- Applications
www.erlang.org/doc/design_principles/applications.html ▾ Erlang ▾
To do this, we **create** an **application** callback module, where we describe how ... If we use systools, the **Erlang**/OTP tools for packaging code (see Releases), the ...

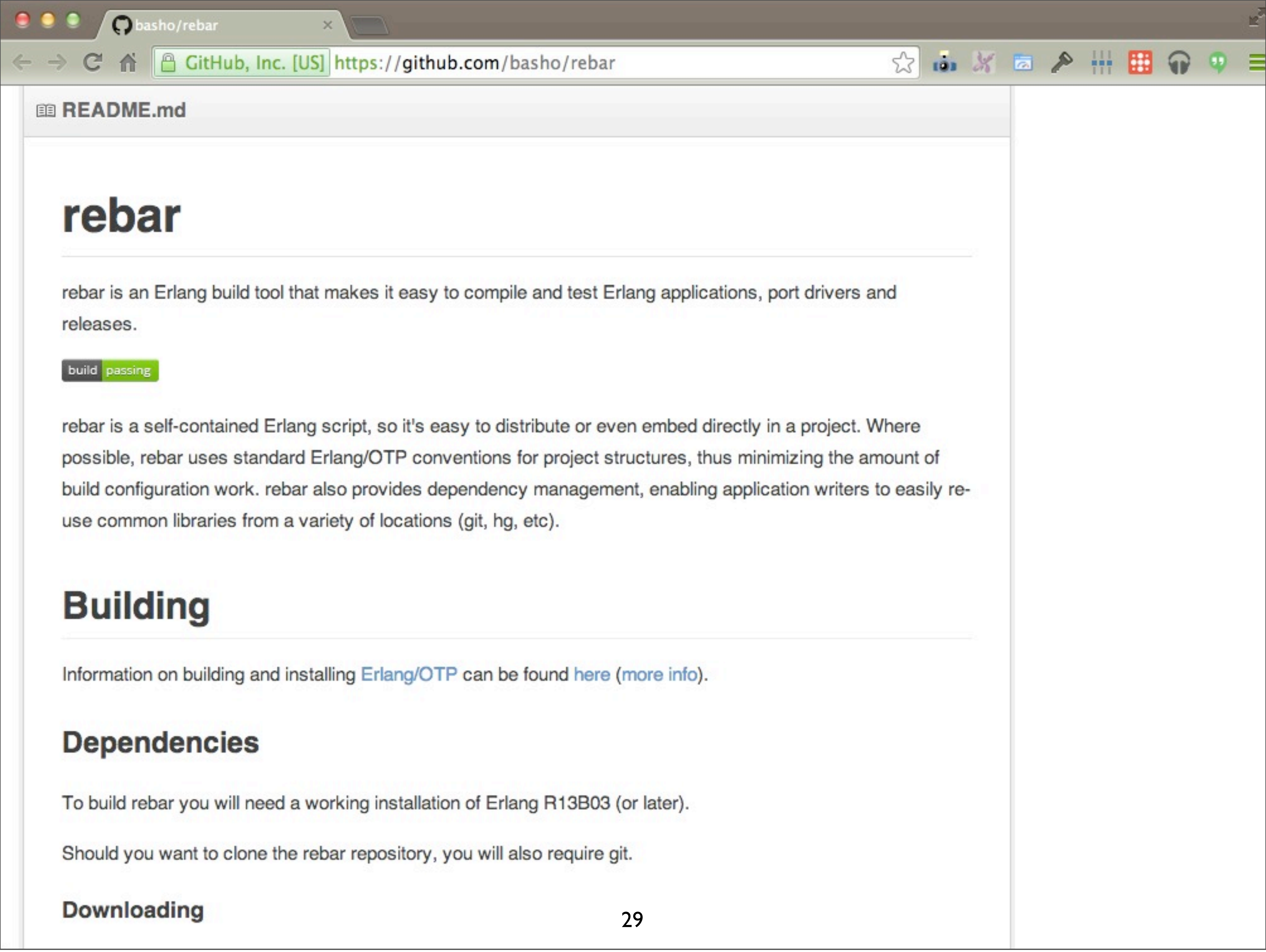
Building OTP Applications | Learn You Some Erlang for Gre...
learnyousomeerlang.com/building-otp-applications ▾
After seeing our whole **application's** supervision tree start at once with a simple function call, we might wonder why we would want to **make** things more ...

Building Applications With OTP | Learn You Some Erlang fo...
learnyousomeerlang.com/building-applications-with-otp ▾
We've not exactly seen how to use them together to **build** applications and tools, though. An **Erlang application** is a group of related code and processes.

basho/rebar · GitHub
https://github.com/basho/rebar ▾
Nov 6, 2013 - README.md. rebar. rebar is an Erlang **build** tool that makes it easy to compile and test **Erlang applications**, port drivers and releases.
You visited this page.

How to run erlang (rebar build) application - Stack Overflow
stackoverflow.com/.../how-to-run-erlang-rebar-build-app... ▾ Stack Overflow ▾
May 21, 2013 - start() -> error_logger:info_msg("Starting app(dev)..~n") ... For quick development, if you just want to ensure your appliction can start, start a shell, ...

Skeptomai | "Don't try to use what you learn from Buddhism t



basho/rebar

GitHub, Inc. [US] https://github.com/basho/rebar

README.md

rebar

rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.

build passing

rebar is a self-contained Erlang script, so it's easy to distribute or even embed directly in a project. Where possible, rebar uses standard Erlang/OTP conventions for project structures, thus minimizing the amount of build configuration work. rebar also provides dependency management, enabling application writers to easily re-use common libraries from a variety of locations (git, hg, etc).

Building

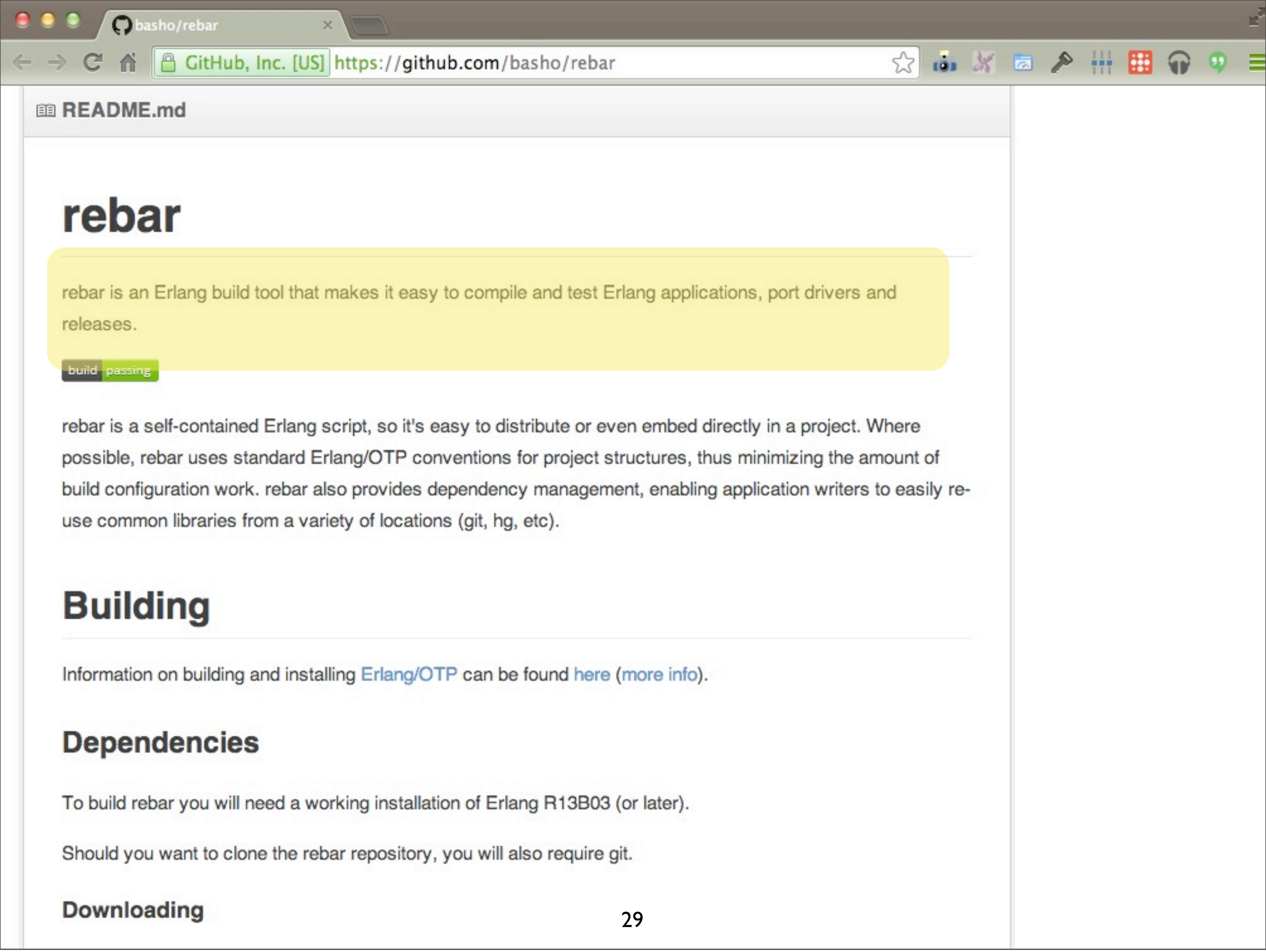
Information on building and installing Erlang/OTP can be found here (more info).

Dependencies

To build rebar you will need a working installation of Erlang R13B03 (or later).

Should you want to clone the rebar repository, you will also require git.

Downloading



rebar

rebar is an Erlang build tool that makes it easy to compile and test Erlang applications, port drivers and releases.

rebar is a self-contained Erlang script, so it's easy to distribute or even embed directly in a project. Where possible, rebar uses standard Erlang/OTP conventions for project structures, thus minimizing the amount of build configuration work. rebar also provides dependency management, enabling application writers to easily re-use common libraries from a variety of locations (git, hg, etc).

Building

Information on building and installing Erlang/OTP can be found here (more info).

Dependencies

To build rebar you will need a working installation of Erlang R13B03 (or later).

Should you want to clone the rebar repository, you will also require git.

Downloading

```
$ git clone git://github.com/rebar/rebar.git
Cloning into 'rebar'...
Resolving deltas: 100% (3633/3633), done.

$ cd rebar

$ ./bootstrap
Recompile: src/rebar
Recompile: src/rebar_abnfc_compiler
Recompile: src/rebar_app_utils
Recompile: src/rebar_appups
. . .
Recompile: src/rebar_xref
==> rebar (compile)
==> rebar (escriptize)
Congratulations! You now have a self-contained script
called "rebar" in your current working directory. Place
this script anywhere in your path and you can use rebar
to build OTP-compliant apps.
$
```

```
$ ./rebar create-app app-id=my-app
==> rebar (create-app)
Writing src/myapp.app.src
Writing src/myapp_app.erl
Writing src/myapp_sup.erl
```

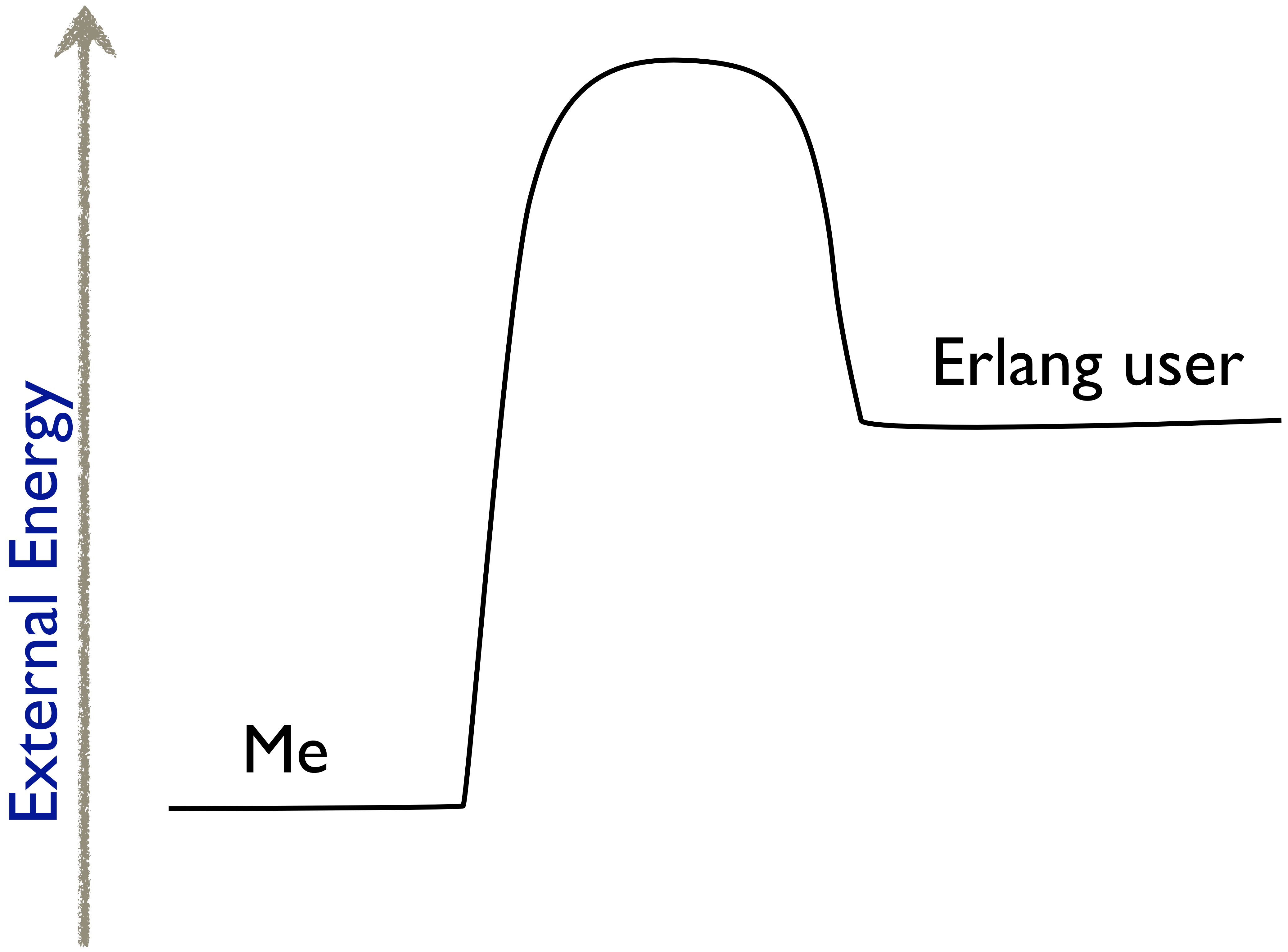
```
$ ./rebar create-app app-id=my-app
==> rebar (create-app)
Writing src/myapp.app.src
Writing src/myapp_app.erl
Writing src/myapp_sup.erl

$ cd my-app
No such file or directory: my-app
```

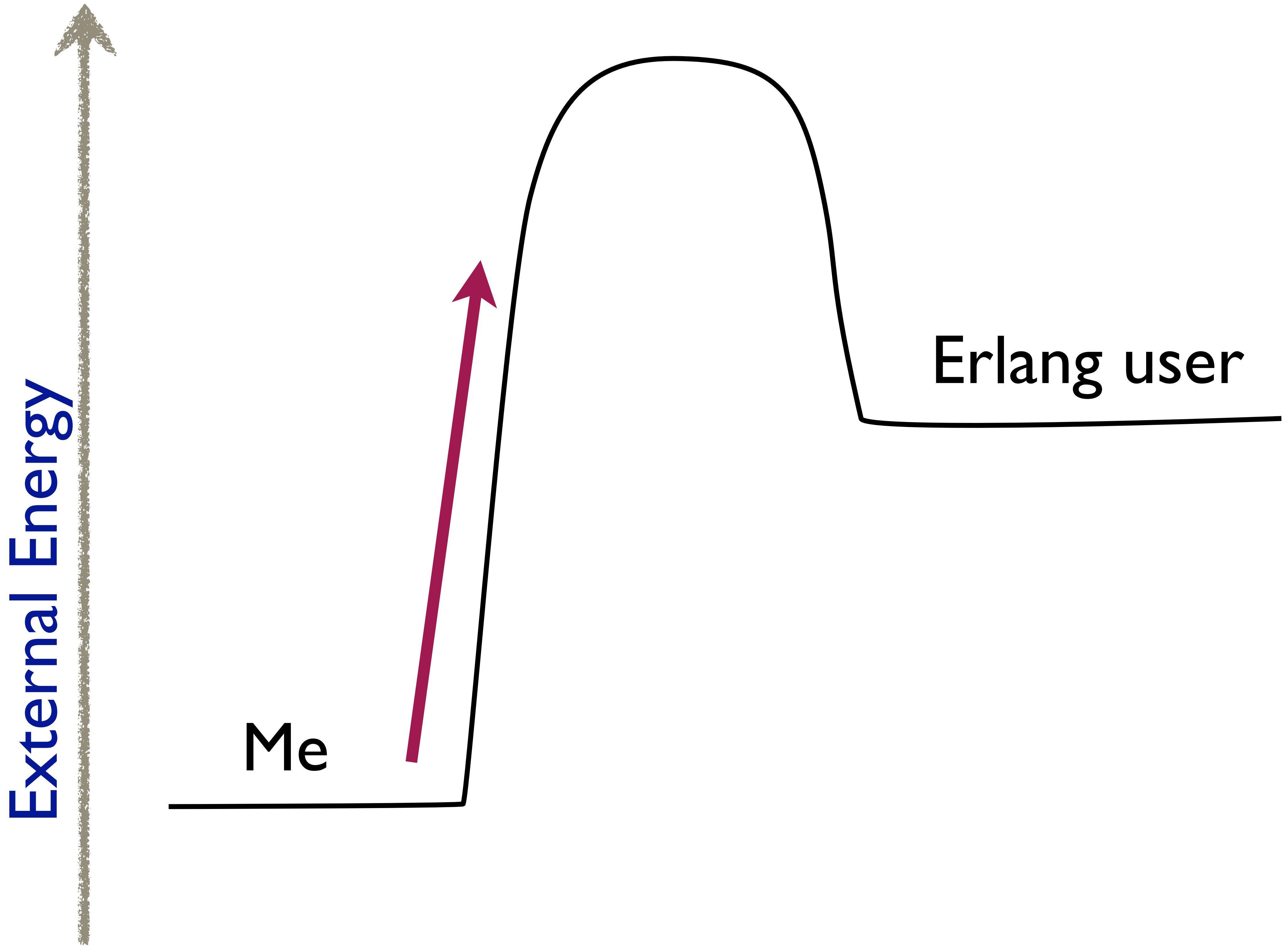
```
$ ./rebar create-app app-id=my-app
==> rebar (create-app)
Writing src/myapp.app.src
Writing src/myapp_app.erl
Writing src/myapp_sup.erl

$ cd my-app
No such file or directory: my-app

$ cd src
$ ls
myapp.app.src              rebar_file_utils.erl
myapp_app.erl              rebar_getopt.erl
myapp_sup.erl              rebar_lfe_compiler.erl
rebar.erl                  rebar_log.erl
rebar_abnfc_compiler.erl   rebar_mustache.erl
rebar_app_utils.erl        rebar_neotoma_compiler.erl
rebar_appups.erl           rebar_otp_app.erl
```

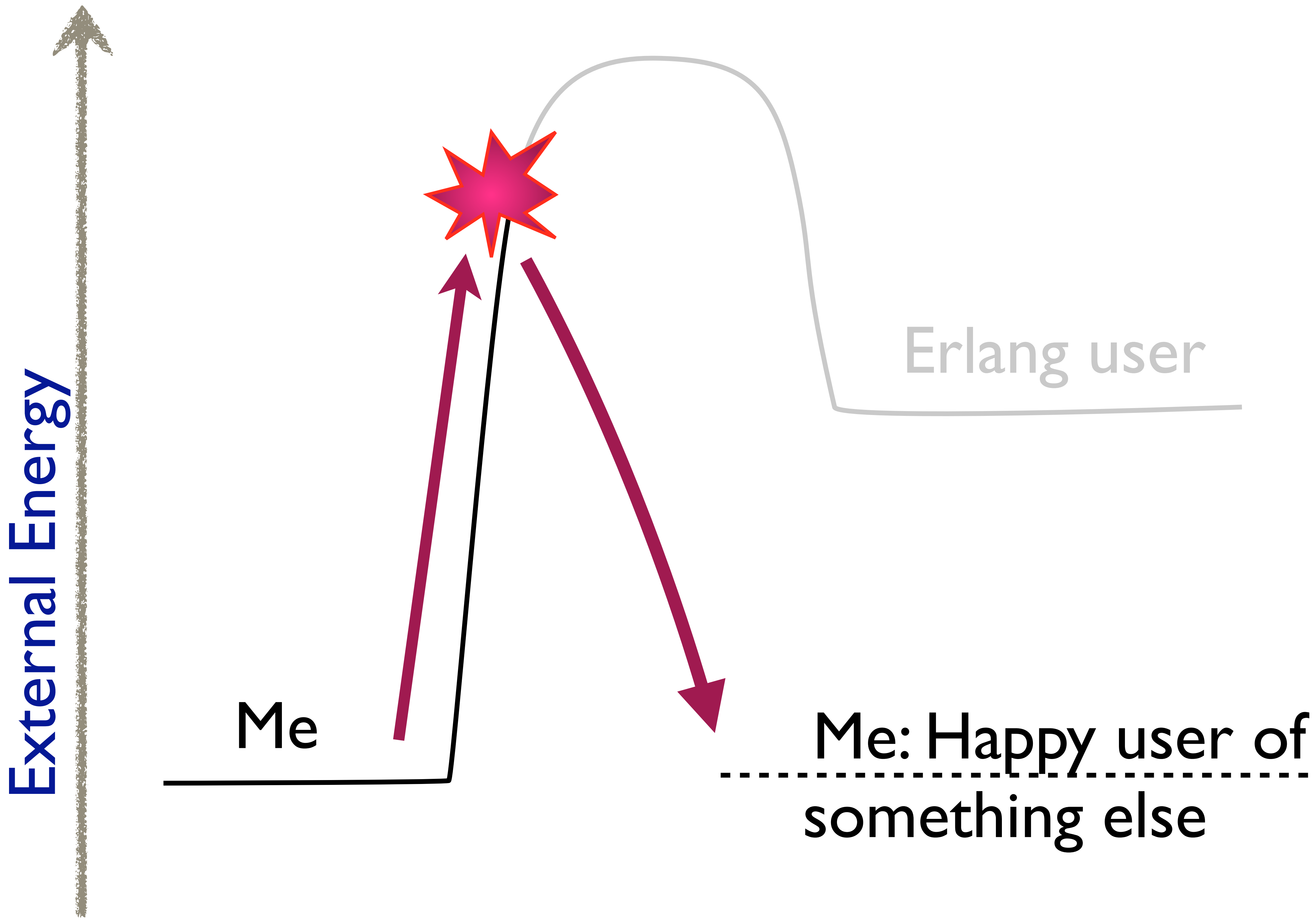
```
$ ./rebar create-app app-id=my_app
==> MyApp (create-app)
Writing src/my_app.app.src
Writing src/my_app_app.erl
Writing src/my_app_sup.erl

$ ./rebar compile
==> MyApp (compile)
Compiled src/my_app_app.erl
Compiled src/my_app_sup.erl

$ erl -pa ebin -s my_app
Erlang/OTP 17 [RELEASE CANDIDATE 1] [erts-6.0]
[source] [64-bit] [smp:4:4] [async-threads:10] [hipe]
[kernel-poll:false]

{"init terminating in do_boot",{undef,[{my_app,start,
[],[]},{init,start_it,1,[]},{init,start_em,1,[]}]}}

Crash dump was written to: erl_crash.dump
init terminating in do_boot ()
```

```
$ erl -pa ebin -s my_app
. . .
{"init terminating in do_boot",{undef,[{my_app,start,
[],[]},{init,start_it,1,[]},{init,start_em,1,[]}]}}

Crash dump was written to: erl_crash.dump
init terminating in do_boot ()
```

- Why are errors displayed as Erlang terms?
- What is the error
  - (remember, I'm new to Erlang)
- What should I do next?


External Energy
Me
Erlang user


External Energy
Me
Erlang user


External Energy
Erlang user
Me
Me: Happy user of
something else

This is just the mechanics

Thinking in Erlang is hard, too…

- Pattern matching
- Recursion
- Higher Order Functions
- Anonymous functions
- Expression-based conditionals (case/if)
- Single Assignment
- Immutability (and the lack for while and for loops)
- Not using objects

(Fred Herbert 2/13/14)

Nested Data Structures

```
#{
  RoomId =>
    #room{
      users=#{
        UserId => [Codes]
      }
    }
}.
```

User joins a room

```
Rooms#{RoomId}#room.users#{UserId} += [NewCode]
```

User joins a room

```
join_room(#{RoomId := Room} = Rooms,
          RoomId, UserId, NewCode) ->
  Rooms#{RoomId := join_room(Room, UserId, NewCode)}.

join_room(#room{users=Users} = Room, UserId, NewCode) ->
  Room#room{users=join_room(Users, UserId, NewCode)};

join_room(#{UserId := Codes} = Users, UserId, NewCode) ->
  Users#{UserId := join_room(Codes, NewCode)}.

join_room(Codes, NewCode) ->
  [NewCode|Codes].
```

Is this the correct design?

Clojure

- get_in
- assoc_in
- update_in

In Erlang

```
update_in(Rooms, [RoomId, #room.users, UserId],
          fun(Codes) -> [NewCode|Codes] end)
```

Haskell Lenses

https://github.com/jlouis/erl-lenses

OTP

I need to write
an application

I need to write an application

OTP!

I need to
handle events

I need to handle events

OTP!

I need to do what is best for my customer

I need to do what is best for my customer

OTP!

I need to
store state

I need to store state

OOP!

I need world peace

I need world peace

OTP!

OTP!

OTP!

OTP!

OTP is Cool, But…

- High ceremony
- Steep learning curve
- Much duplication
  - cut and paste code
  - API vs. handler

Simple Problem

- Need to parse a configuration file and access its data throughout the application life cycle

Config Server

```
-module(config).

-behaviour(gen_server).

%% API
-export([start_link/0]).

%% gen_server callbacks
-export([init/1, handle_call/3, handle_cast/2, handle_info/2,
         terminate/2, code_change/3]).

-define(SERVER, ?MODULE).

start_link() ->
    gen_server:start_link({local, ?SERVER}, ?MODULE, [], []).

init([]) ->
    {ok, parse_config()}.

handle_call(_Request, _From, State) ->
    Reply = ok,
    {reply, Reply, State}.

handle_cast(_Msg, State) ->
    {noreply, State}.

handle_info(_Info, State) ->
    {noreply, State}.

terminate(_Reason, _State) ->
    ok.

code_change(_OldVsn, State, _Extra) ->
    {ok, State}.
```

Config using Agents

```
(def config (agent (parse-config)))

(await (send config ...))

@config
```

Futures

Futures

```
(new Future(getUsers))
  .onSuccess(...)
  .onFailure(...)
```

Erlang mismatch

- Callback soup
- Conflated error handling

They want to start a computation, asynchronously, and later read its value back

.NET Task Parallel Library

```
task = new Task(action)
// some computation
task.Wait()
```

In Erlang

```
From = self(),
Pid  = spawn_link(fun() ->
          From ! {self(), Action()}
       end),

% some computation

receive
  {Pid,Res} -> Res
end;
```

```
task(Action) ->
  From = self(),
  Ref  = erlang:make_ref(),
  spawn_link(fun() ->
    case (catch Action()) of
      {'EXIT', Why} ->
        From ! {Ref, {error, Why}};
      Reply ->
        From ! {Ref, {ok, Reply}}
    end
  end),
  Ref.
```

```erlang
wait(Ref) when is_reference(Ref) ->
  receive
    {Ref, {error, Why}} -> error(Why);
    {Ref, {ok, Reply}}  -> Reply
  end.
```

We can implement tasks in about 15 LOC

Can we expect someone with 2 weeks of Erlang experience to write this code?

The Erlang Gap

Lower The Barriers

Lower The Barriers

- My First Erlang Program
  - should take 10 minutes from Erlang install to success
  - recommended tutorials, videos, and downloads to point the way

Lower The Barriers

- Error messages should be aimed at humans, not file:consult/1
- {error, enoent} is cool, but which file?
- (Maybe include lager by default?)

Lower The Barriers

- Provide modern abstractions
  - such as Clojure's get_in, assoc_in, update_in
  - built-in implementations of things such as agents, tasks, (reactive APIs, etc…)

Lower The Barriers

Think like a newcomer

Lower The Barriers

Lower The Barriers

Share the Love