

Get it done.
One event at a time.



How I learned to stop worrying and
love EventMachine.

Who Am I

- Dan Sinclair (dan@aiderss.com)
- AideRSS code monkey



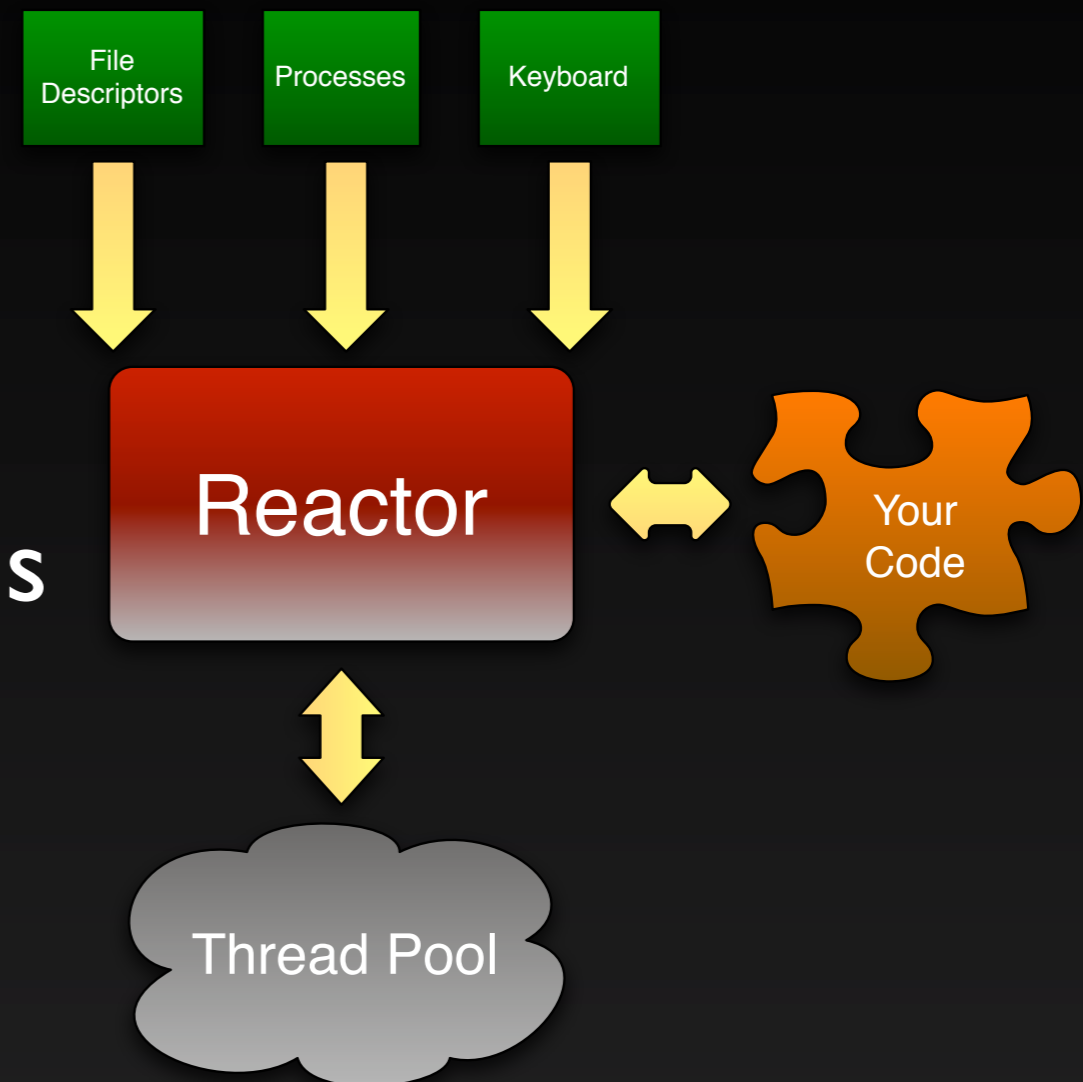
Overview

- What
- Why
- How
- Gotchas



What is EventMachine

- Reactor pattern
- Protocol implementations
- Thread pool



Why

- Performance
- Simplicity
- Scalability



How

- Basics
- Timers
- Scheduling
- Lightweight Concurrency
- Clients / Servers



Set 'em up

- EM.run
- EM.stop aka EM.stop_event_loop
- EM.reactor_running?
- Initialization
- Blocking



EM.run

```
require 'rubygems'  
require 'eventmachine'  
  
puts "Start"  
EM.run do  
  puts "Init"  
  EM.add_timer(1) do  
    puts "Quitting"  
    EM.stop_event_loop  
  end  
end  
puts "Done"
```

```
titania:examples dj2$ ruby run.rb  
Start  
Init  
Quitting  
Done
```



EM.reactor_running?

```
require 'rubygems'  
require 'eventmachine'
```

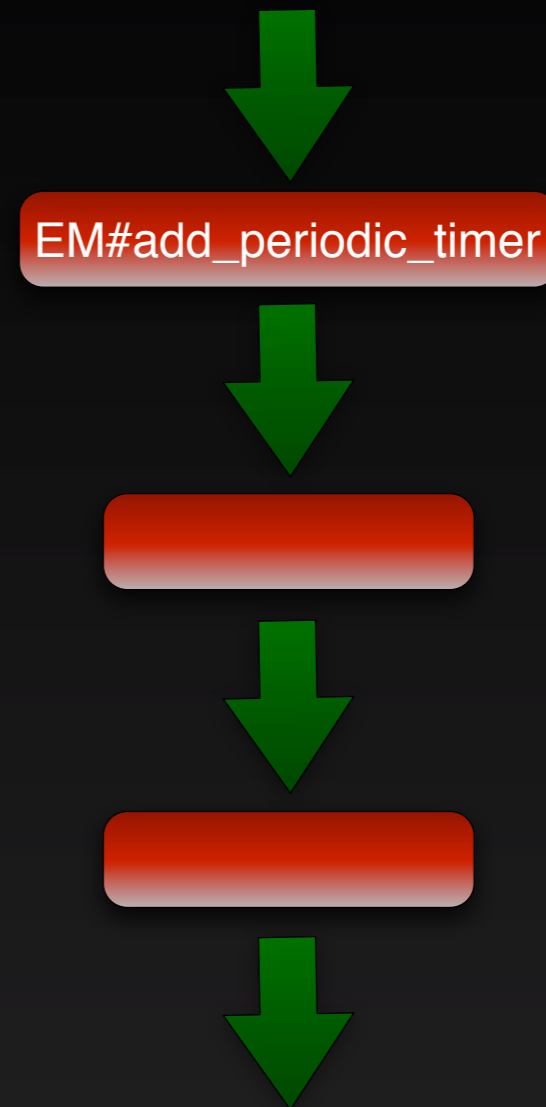
```
puts EM.reactor_running?  
EM.run do  
  puts EM.reactor_running?  
  EM.next_tick do  
    puts EM.reactor_running?  
    EM.stop  
  end  
end  
puts EM.reactor_running?
```

```
titania:EventMachine dj2$ ruby running.rb  
false  
true  
true  
false
```



On a Schedule

- EM.add_periodic_timer(time)
- EM.add_timer(time)



EM.add_ *_? _timer

```
require 'rubygems'  
require 'eventmachine'
```

```
EM.run do  
  EM.add_periodic_timer(1) do  
    puts "Tick..."  
  end  
  EM.add_timer(5) do  
    puts "BOOM"  
    EM.stop_event_loop  
  end  
end
```

```
titania:examples dj2$ ruby digging_in.rb  
Tick...  
Tick...  
Tick...  
Tick...  
BOOM
```



Do it Later

- EM.next_tick
- Synchronous



EM.next_tick

```
require 'rubygems'  
require 'eventmachine'
```

```
EM.run do  
  EM.add_periodic_timer(1) do  
    puts "Hai"  
  end  
  EM.add_timer(5) do  
    EM.next_tick do  
      EM.stop_event_loop  
    end  
  end  
end
```

```
titania:examples dj2$ ruby next_tick.rb
```

```
Hai
```

```
Hai
```

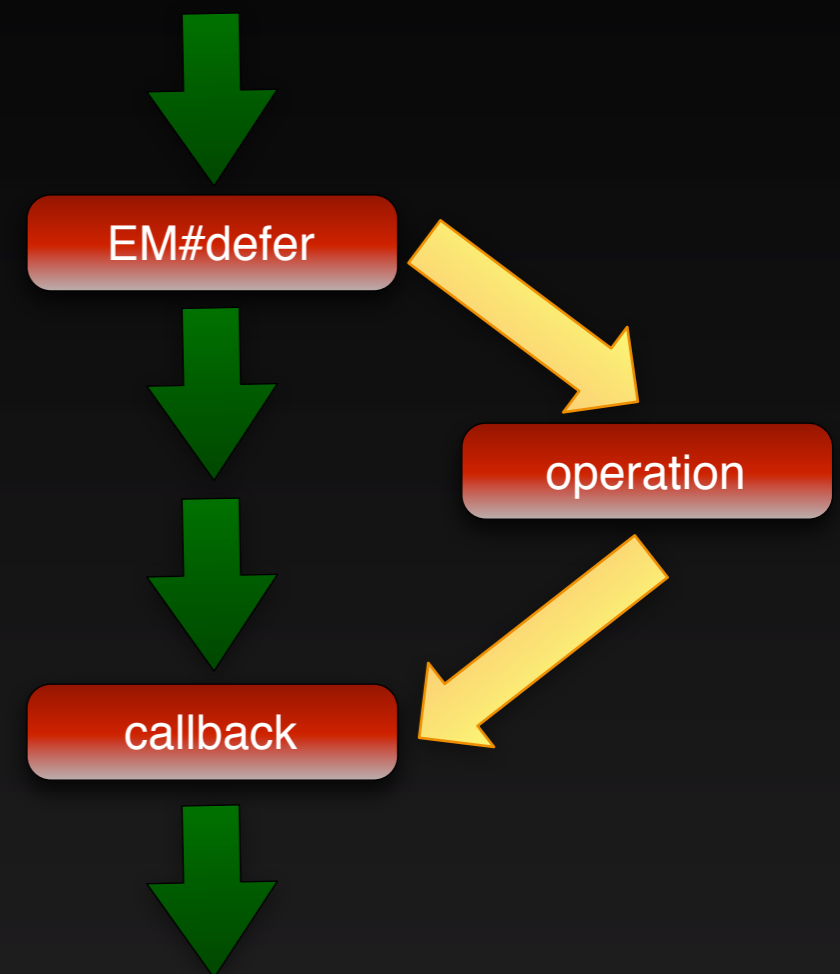
```
Hai
```

```
Hai
```



Do it Now(ish)

- EM.defer(operation, callback)
- Asynchronous operation
- Synchronous callback



EM.defer

```
require 'rubygems'  
require 'eventmachine'  
require 'thread'
```

```
EM.run do  
  EM.add_timer(2) do  
    puts "Main #{Thread.current}"  
    EM.stop_event_loop  
  end  
  EM.defer(proc do  
    puts "Defer #{Thread.current}"  
  end)  
end
```

```
titania:examples dj2$ ruby defer.rb  
Defer #<Thread:0x5637f4>  
Main #<Thread:0x35700>
```



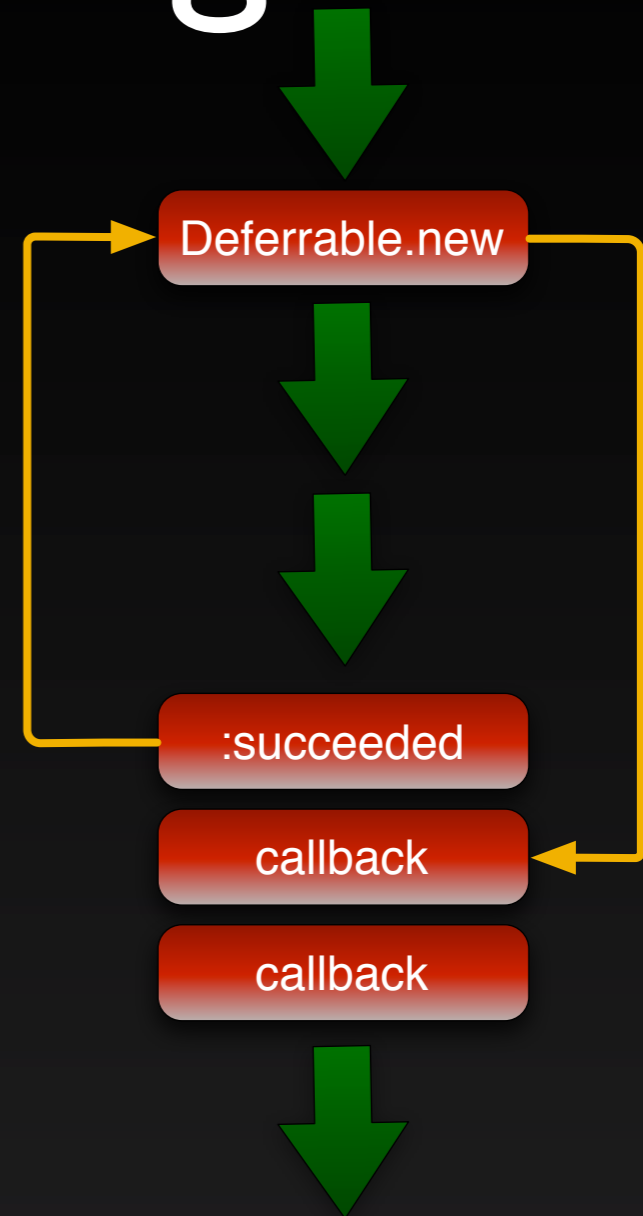
next_tick vs defer

- Long running tasks
- Simultaneous tasks



Procrastinating

- EM::Deferrable
- EM::DefaultDeferrable
- callback / errback



EM::Deferrable

```
require 'rubygems'  
require 'eventmachine'
```

```
class MyDeferrable  
  include EM::Deferrable  
  def go(str)  
    puts "Go #{str} go"  
  end  
end
```

```
EM.run do  
  df = MyDeferrable.new  
  df.callback do |x|  
    df.go(x)  
    EM.stop  
  end  
  EM.add_timer(1) do  
    df.set_deferred_status :succeeded, "SpeedRacer"  
  end  
end
```

```
titania:EventMachine dj2$ ruby deferrable.rb  
Go SpeedRacer go
```



callback / errback

```
require 'rubygems'  
require 'eventmachine'
```

```
EM.run do
```

```
  df = EM::Protocols::HttpClient.request(:host => 'postrank.com',  
                                         :request => '/')
```

```
  df.callback do |response|
```

```
    puts "Succeeded: #{response[:status]}"
```

```
    EM.stop
```

```
  end
```

```
  df.errback do |response|
```

```
    puts "ERROR: #{response[:status]}"
```

```
    EM.stop
```

```
  end
```

```
end
```

```
titania:EventMachine dj2$ ruby http_client.rb  
Succeeded: 301
```



EM::DefaultDeferrable

```
require 'rubygems'  
require 'eventmachine'
```

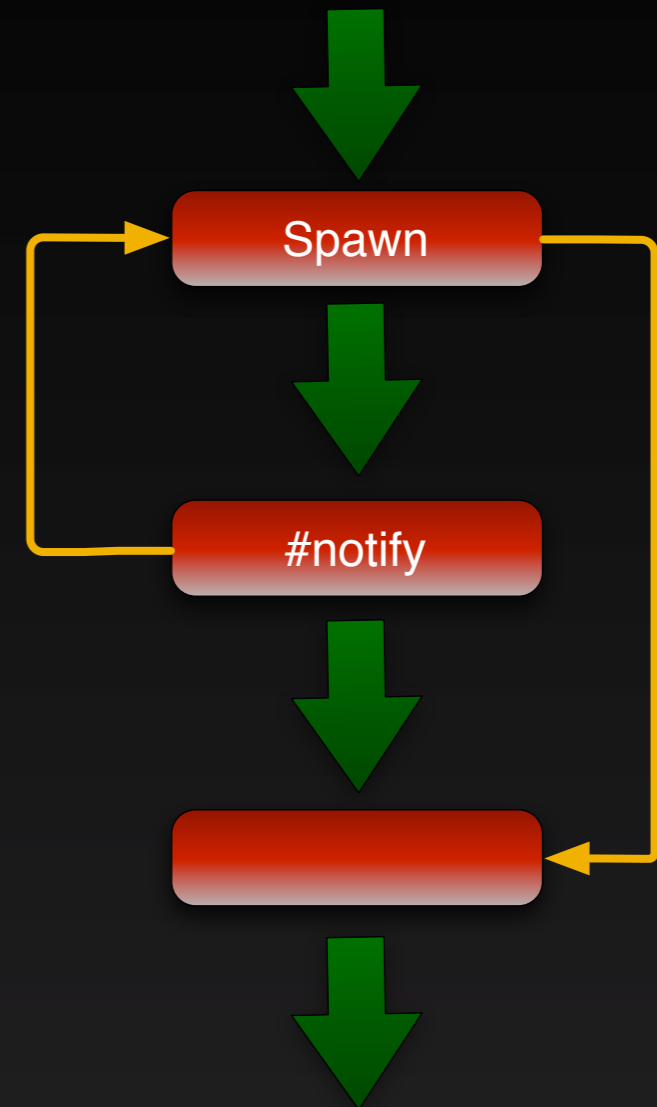
```
EM.run do  
  df = EM::DefaultDeferrable.new  
  df.callback do |x|  
    puts "got #{x}"  
  end  
  df.callback do |x|  
    EM.stop  
  end  
  EM.add_timer(1) do  
    df.set_deferred_status :succeeded, "monkeys"  
  end  
end
```

```
titania:EventMachine dj2$ ruby default_deferrable.rb  
got monkeys
```



Spawning

- EM.spawn
- EM::SpawnedProcess
- Not an OS process



EM.spawn

```
require 'rubygems'  
require 'eventmachine'
```

```
EM.run do  
  s = EM.spawn do |val|  
    puts "Received #{val}"  
  end  
  EM.add_timer(1) do  
    s.notify "hello"  
  end  
  EM.add_periodic_timer(1) do  
    puts "Periodic"  
  end  
  EM.add_timer(3) do  
    EM.stop  
  end  
end
```

```
Rei:EventMachine dj2$ ruby spawn.rb  
Periodic  
Received hello  
Periodic
```



Speak to Me

- EM.start_server
- EM.start_unix_domain_server



EM.start_server

```
require 'rubygems'  
require 'eventmachine'
```

```
module Echo  
  def receive_data data  
    send_data data  
  end  
end
```

```
EM.run do  
  EM.start_server "0.0.0.0", 10000, Echo  
end
```

```
Rei:~ dj2$ telnet localhost 10000  
Trying 127.0.0.1...  
Connected to localhost.  
Escape character is '^]'.  
helo  
helo  
goodbye cruel world  
goodbye cruel world
```



Same as ...

```
require 'rubygems'  
require 'eventmachine'
```

```
class Echo < EventMachine::Connection  
  def receive_data data  
    send_data data  
  end  
end
```

```
EM.run do  
  EM.start_server "0.0.0.0", 10000, Echo  
end
```



Passing the Buck

```
class Pass < EM::Connection
  attr_accessor :a
  def receive_data(data)
    send_data "#{@a} " +
              "#{data.chomp}"
  end
end

EM.run do
  EM.start_server("127.0.0.1", 10000, Pass) do |conn|
    conn.a = "Goodbye"
  end
end
```

```
titania:~ dj2$ telnet localhost 10000
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
mostly cruel world
Goodbye mostly cruel world
```



The Chicago Manual

- `post_init`
- `connection_completed`
- `receive_data(data)`
- `send_data(data)`
- `unbind`
- `connection_close`



Speak to You

- EM.connect
- EM.connect_unix



```
require 'rubygems'
require 'eventmachine'
```

EM.connect

```
class Connector < EM::Connection
```

```
  def post_init
```

```
    puts "Getting /"
```

```
    send_data "GET / HTTP/1.1\r\nHost: MagicBob\r\n\r\n"
```

```
  end
```

```
  def receive_data(data)
```

```
    puts "Received #{data.length} bytes"
```

```
    close_connection_after_writing
```

```
  end
```

```
  def unbind
```

```
    puts "Terminated"
```

```
  end
```

```
end
```

```
EM.run do
```

```
  EM.connect "www.postrank.com", 80, Connector
```

```
end
```

```
titania:EventMachine dj2$ ruby connect.rb
```

```
Getting /
```

```
Received 1448 bytes
```

```
Terminated
```



Doing it HTTP Style

- `evma_httpserver`
- Not RFC compliant



EM::HttpServer

```
require 'rubygems'  
require 'eventmachine'  
require 'evma_httpserver'
```

```
class HttpServer < EM::Connection  
  include EM::HttpServer
```

```
  def process_http_request  
    response = EM::DelegatedHttpResponse.new(self)  
    response.status = 200  
    response.content_type 'text/html'  
    response.content = '<center><h1>Wham-o</h1></center>'  
    response.send_response  
  end  
end
```

```
EM.run do  
  EM.start_server '0.0.0.0', 8080, HttpServer  
end
```



Talking to the Rabbit

- AMQP extension
- Callback based
- Reactor running



AMQP

```
require 'rubygems'  
require 'mq'
```

```
EM.run do
```

```
  connection = AMQP.connect(:user => 'metrics', :pass => 'password',  
                           :host => 'rei-lab', :vhost => '/attention')
```

```
  channel = MQ.new(connection)
```

```
  xchange = MQ::Exchange.new(channel, :fanout, 'test-xchange')
```

```
  q = MQ::Queue.new(channel, 'test-queue', :durable => true)
```

```
  q.bind(xchange)
```

```
  q.subscribe do |header, msg|
```

```
    puts msg
```

```
  end
```

```
end
```



Gotchas

- Thread safety
->>>6
- Connection independence
- Thread performance



Questions / Comments

- dan@aiderss.com



Resources

- <http://rubyeventmachine.com>
- <http://groups.google.com/group/eventmachine/>
- <http://github.com/eventmachine/eventmachine>

