# Node.js

## Should ruby developers care?

09.02.2011

# About

- Felix Geisendörfer
- 23 years
- Berlin, Germany


Core Contributor

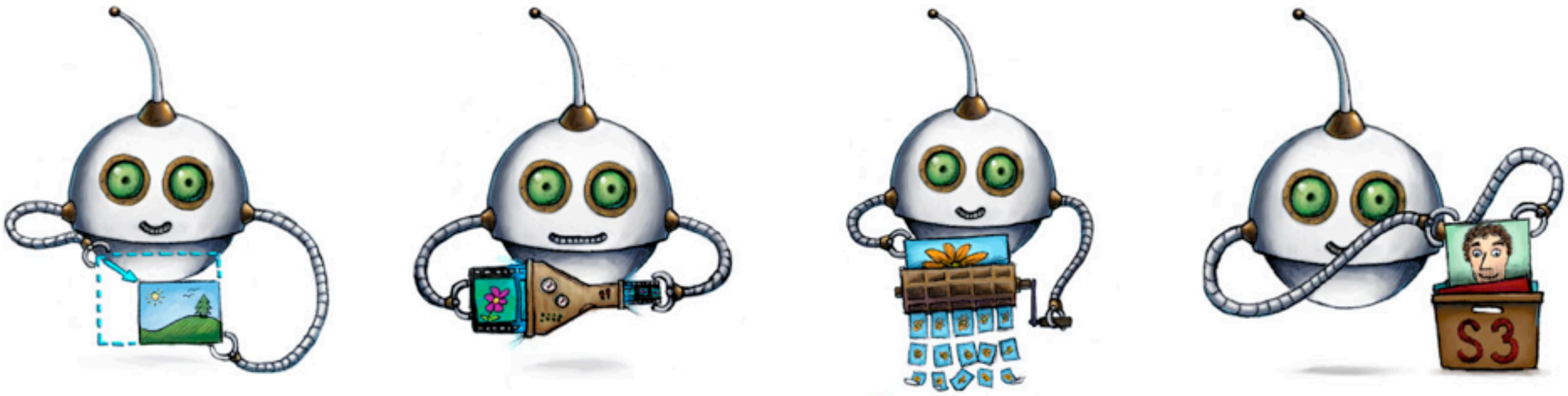
&

Module Author


node-mysql

node-formidable

Transloadit




Resize images

Encode videos

Extract thumbnails

Store in S3

File uploading & processing as an infrastructure service for web & mobile applications.

# About this talk


Node's goal is to provide an easy way to build scalable network programs.

# Node.js

- Created by Ryan Dahl
- Google's V8 JavaScript engine (no DOM)
- Module system, I/O bindings, Common Protocols

# Installing

```
$ git clone \
git://github.com/ry/node.git
$ cd node
$ ./configure
$ make install
```

# Hello World

```
$ cat test.js
console.log('Hello World');

$ node test.js
Hello World
```

# Ingredients

libeio

libev

c-ares

V8

http_parser

# Philosophy

- Just enough core-library to do I/O
- Non-blocking
- Close to the underlaying system calls

# Non-blocking I/O

# Blocking I/O

```
var a = db.query('SELECT A');
console.log('result a:', a);
```

```
var b = db.query('SELECT B');
console.log('result b:', b);
```

Time = SUM(A, B)

# Non-Blocking I/O

```
db.query('SELECT A', function(result) {
  console.log('result a:', result);
});

db.query('SELECT B', function(result) {
  console.log('result b:', result);
});
```

Time = MAX(A, B)

# Non-Blocking I/O

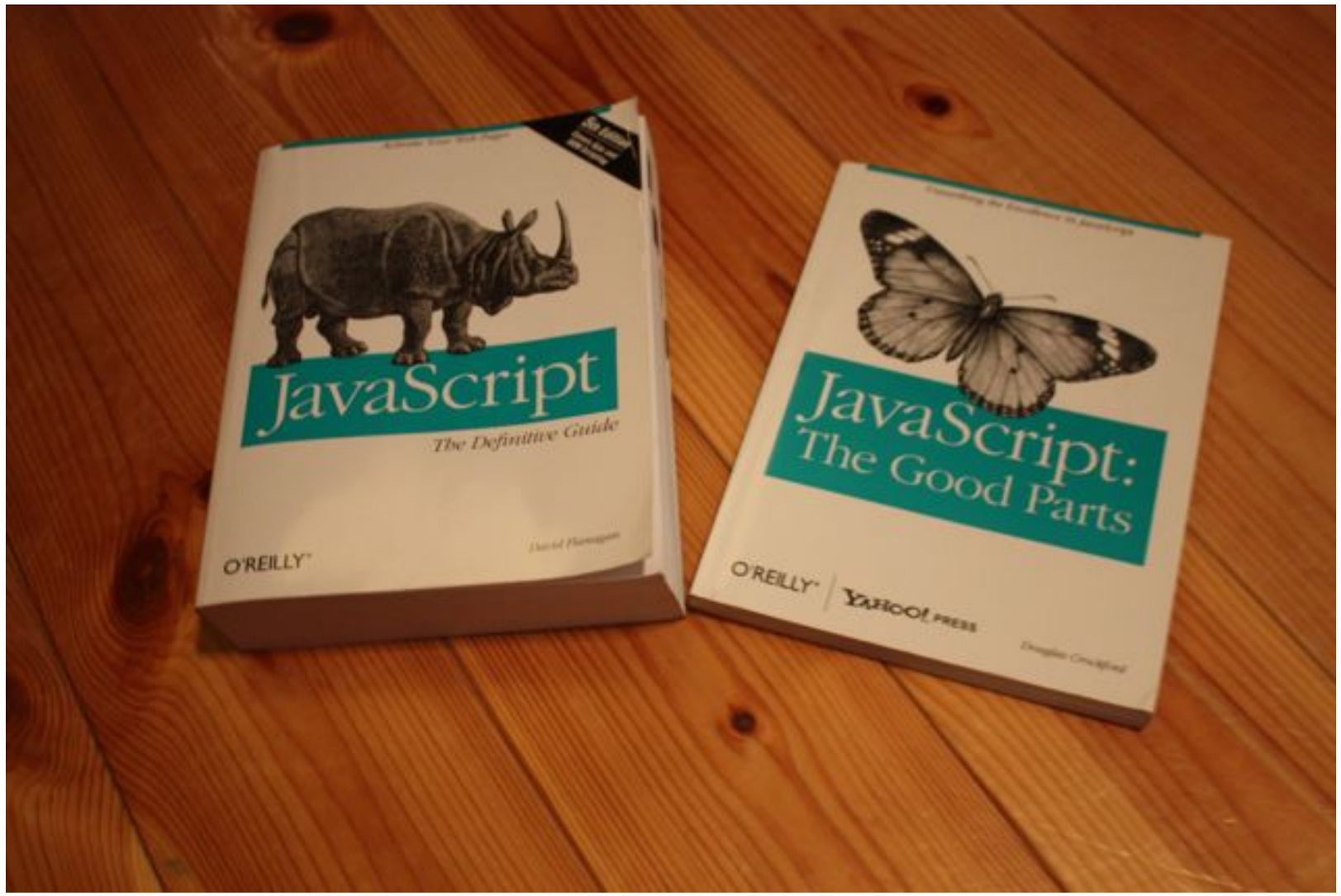
- Offload most things to the kernel and poll for changes (select, epoll, kqueue, etc.)

- Thread pool for anything else

# Why JavaScript?

3 Reasons

# #1 - The good parts



V8 (Chrome)

SpiderMonkey (Firefox)

JavaScriptCore (Safari)

# #2 - JS Engine Wars

Chakra (IE9)

Carakan (Opera)

# #3 - No I/O in the core

# What about event machine?

- Can't use blocking I/O libraries
- No code sharing between client and server
- Might be a valid choice if integrating with existing ruby code

# Frameworks

# Frameworks

- Express (popular)
- Fab (revolutionary)

# Rails still kicks node's ass in terms of productivity

# Suitable Applications

- Singe-page apps
- Real time
- Crawlers

# More Applications

- Process monitoring
- File uploading
- Streaming

# What about all those callbacks?

```
db.query('SELECT A', function() {
  db.query('SELECT B', function() {
    db.query('SELECT C', function() {
      db.query('SELECT D', function() {
        // ouch
      });
    });
  });
});
```

# You are holding it wrong!

# Nested Callbacks

- Function does too many things, break it up
- Use an sequential abstraction library
- Could be a sign that ruby is a better choice

# Syntactical Sugar

# CoffeeScript

```coffeescript
# Assignment:
number   = 42
opposite = true

# Conditions:
number = -42 if opposite

# Functions:
square = (x) -> x * x

# Arrays:
list = [1, 2, 3, 4, 5]

# Objects:
math =
  root:   Math.sqrt
  square: square
  cube:   (x) -> x * square x

# Splats:
race = (winner, runners...) ->
  print winner, runners

# Existence:
alert "I knew it!" if elvis?

# Array comprehensions:
cubes = (math.cube num for num in list)
```

```javascript
var cubes, list, math, num, number, opposite, race, square;
var __slice = Array.prototype.slice;
number = 42;
opposite = true;
if (opposite) {
  number = -42;
}
square = function(x) {
  return x * x;
};
list = [1, 2, 3, 4, 5];
math = {
  root: Math.sqrt,
  square: square,
  cube: function(x) {
    return x * square(x);
  }
};
race = function() {
  var runners, winner;
  winner = arguments[0], runners = 2 <= arguments.length ?
__slice.call(arguments, 1) : [];
  return print(winner, runners);
};
if (typeof elvis != "undefined" && elvis !== null) {
  alert("I knew it!");
}
cubes = (function() {
  var _i, _len, _results;
  _results = [];
  for (_i = 0, _len = list.length; _i < _len; _i++) {
    num = list[_i];
    _results.push(math.cube(num));
  }
  return _results;
})();
```

run: cubes

# Ready for production?

# Ready for production?

- 0.4 to be released this month
- API has settled down for most parts
- Very few bugs, but YMMV

# Things that suck

# Things that suck

- Stack traces are lost at the event loop boundary
- Utilizing multiple cores requires multiple processes
- GC can be problematic

# Speed

# Speed

```
var http = require('http')
var b = new Buffer(1024*1024);

http.createServer(function (req, res) {
  res.writeHead(200);
  res.end(b);
}).listen(8000);
```

by Ryan Dahl

# Speed

100 concurrent clients
1 megabyte response

node 822 req/sec
nginx 708
thin 85
mongrel 4

(bigger is better)

by Ryan Dahl

# Speed

- NGINX peaked at 4mb of memory
- Node peaked at 60mb
- Very special circumstances

by Ryan Dahl

# Questions?

@felixge / felix@debuggable.com

# Bonus Slides!

# Buffers

# Buffers

- Raw C memory allocation
- Similar to an array of integers
- Can be converted to and from JS strings

# Buffers

```
buffer.write('Hello Buffer');

buffer.toString('utf-8');

buffer[0] = 23;

buffer.slice(10, 20);
```

# Reverse Hello World

```
$ cat reverse_hello.js
var net = require('net');
net.createServer(function(socket) {
  socket.on('data', function(buffer) {
    console.log(buffer);
  });
}).listen(0x27C3);

$ node reverse_hello.js &
[1] 3523
$ echo "hello world" | nc localhost 10179
<Buffer 68 65 6c 6c 6f 20 77 6f 72 6c 64 0a>
```

# Streams

*"Streams are to time as arrays are to space."*

-- Jed Schmidt @ JSConf.eu

# Streams

```
stream.write(new Buffer([0x27, 0xC3]);

stream.pause();

stream.resume();

stream.destroy();
```

# Streams

```
stream
  .on('drain', function() {})
  .on('data', function(buffer) {})
  .on('end', function() {});
```

# Streams

```
$ cat echo.js
var net = require('net');
net.createServer(function(socket) {
   socket.pipe(socket);
}).listen(0x27C3);

$ node echo.js &
[1] 6207
$ nc 127.0.0.1 10179
Hi, how are you?
Hi, how are you?
```

# Questions?

@felixge / felix@debuggable.com