node.js

A quick tour

Felix Geisendörfer

20.04.2011 (v6)

@felixge

Twitter / GitHub / IRC

About

- Felix Geisendörfer
- Berlin, Germany
- Debuggable Ltd

Core Contributor

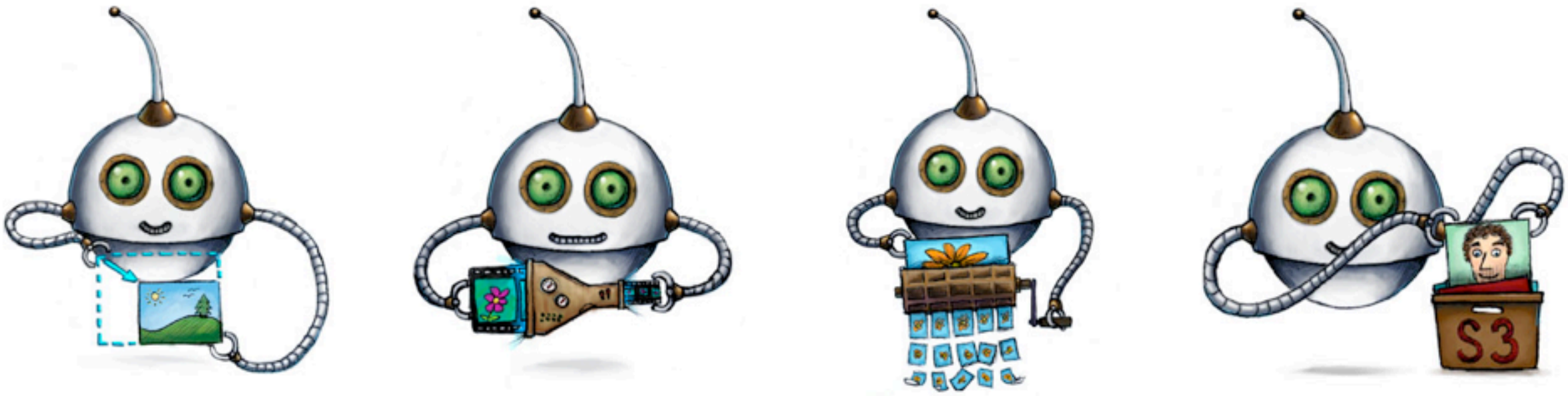
&

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.

Audience?

Server-side JavaScript

Server-side JavaScript

- Netscape LiveWire (1996)
- Rhino (1997)
- 50+ plattforms since then

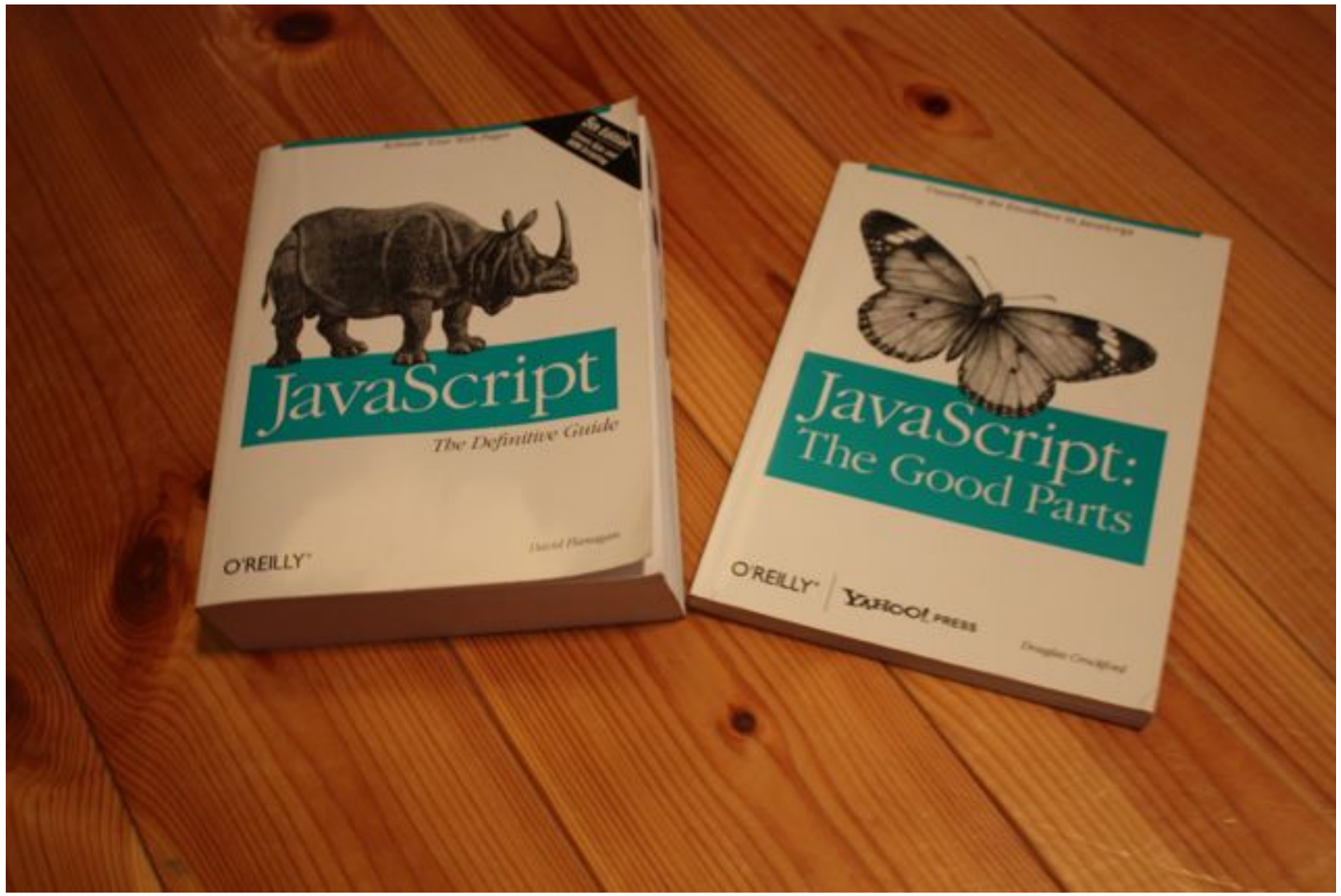
What was the problem?

- Slow engines
- Lack of interest in JS until ~2005
- Better competing plattforms

Why JavaScript?

3 Reasons

#1 - The good parts



V8 (Chrome)

SpiderMonkey (Firefox)

JavaScriptCore (Safari)

#2 - JS Engine Wars

Chakra (IE9)

Carakan (Opera)

check out:
arewefastyet.com

#3 - No I/O in the core

node.JS

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

Single Threaded

```
var a = [];
function first() {
  a.push(1);
  a.push(2);
}

function second() {
  a.push(3);
  a.push(4);
}

setTimeout(first, 10);
setTimeout(second, 10);
```

Single Threaded

```
var a = [];
function first() {
   a.push(1);
   a.push(2);
}

function second() {
   a.push(3);
   a.push(4);
}

setTimeout(first, 10);
setTimeout(second, 10);
```

- [1, 2, 3, 4] ?
- [3, 4, 1, 2] ?
- [1, 3, 2, 4] ?
- BAD_ACCESS ?

Single Threaded

```
var a = [];
function first() {
   a.push(1);
   a.push(2);
}

function second() {
   a.push(3);
   a.push(4);
}

setTimeout(first, 10);
setTimeout(second, 10);
```

- **[1, 2, 3, 4]**
- **[3, 4, 1, 2]**
- ~~[1, 3, 2, 4]~~
- ~~BAD_ACCESS~~

API Overview

CommonJS Modules

```
$ cat hello.js
exports.world = function() {
  return 'Hello World';
};
```

```
$ cat main.js
var hello = require('./hello');
console.log(hello.world());
```

```
$ node main.js
Hello World
```

Child processes

```
$ cat child.js
var spawn = require('child_process').spawn;
var cmd = 'echo hello; sleep 1; echo world;';

var child = spawn('sh', ['-c', cmd]);
child.stdout.on('data', function(chunk) {
  console.log(chunk.toString());
});

$ node child.js
"hello\n"
# 1 sec delay
"world\n\n"
```

Http Server

```
$ cat http.js
var http = require('http');
http.createServer(function(req, res) {
  setTimeout(function() {
    res.writeHead(200);
    res.end('Thanks for waiting!');
  }, 1000);
}).listen(4000);

$ curl localhost:4000
# 1 sec delay
Thanks for waiting!
```

Tcp Server

```
$ cat tcp.js
var tcp = require('tcp');
tcp.createServer(function(socket) {
  socket
    .on('connect', function() {
      socket.write("Hi, How Are You?\n> ");
    })
    .on('data', function(data) {
      socket.write(data);
    });
}).listen(4000);

$ nc localhost 4000
Hi, How Are You?
> Great!
Great!
```

DNS

```
$ cat dns.js
var dns = require('dns');
dns.resolve('nodejs.org', function(err, addresses) {
 console.log(addresses);
});

$ node dns.js
[ '8.12.44.238' ]
```

Watch File

```
$ cat watch.js
var fs = require('fs');
fs.watchFile(__filename, function() {
  console.log('You changed me!');
  process.exit(0);
});

$ node watch.js
# edit watch.js
You changed me!
```

And much more

- UDP
- Crypto
- Assert
- Buffer
- VM
- EcmaScript5

...

Suitable Applications

Suitable Applications

- Singe-page apps
- Real time
- Crawlers

More Applications

- Async chaining of unix tools
- Streaming
- File uploading

Interesting projects

Package management

Web Frameworks

- Express.js (Sinatra clone)

- Fab.js (Mind-bending & awesome)

Socket.IO

```
var http = require('http');
var io = require('socket.io');

var server = http.createServer();
server.listen(80);

var socket = io.listen(server);
socket.on('connection', function(client){
  client.send('hi');

  client.on('message', function(){ … })
  client.on('disconnect', function(){ … })
});
```

Server

Socket.IO

```
<script src="http://{node_server_url}/socket.io/socket.io.js" />
<script>
  var socket = new io.Socket({node_server_url});
  socket.connect();
  socket.on('connect', function(){ … })
  socket.on('message', function(){ … })
  socket.on('disconnect', function(){ … })
</script>
```

Client

Socket.IO

- WebSocket
- Adobe® Flash® Socket
- AJAX long polling
- AJAX multipart streaming
- Forever Iframe
- JSONP Polling

Chooses most capable transport at runtime!

1600+ modules in npm

Ready for production?

Ready for production?

- Handled over 100.000 file uploads
- Several TB of data
- No bugs, Memory usage betwen 30 - 80 MB

Our experience at Transloadit.com

Ready for production?

- 0.4.6 was released on April 13th

- 0.4.x API is stable

- 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

- V8: 1.9 GB heap limit / GC can be problematic

Community

Benevolent Dictator For Life

Ryan Dahl

Community

- Mailing list (nodejs, nodejs-dev)
- IRC (#node.js)

Hosting

More information

nodeguide.com

Meet the community

nodecamp.de

June 11 - 12

Cologne, Germany

Tickets available 12pm CET tomorrow

Questions?

@felixge / felix@debuggable.com

Bonus Slides

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

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
- Consider using a JS language extension

Nested Callbacks

```
err1, result1 = db.query! 'SELECT A'
err2, result2 = db.query! 'SELECT B'
err3, result3 = db.query! 'SELECT C'
err4, result4 = db.query! 'SELECT D'
```

Kaffeeine

Questions?

@felixge / felix@debuggable.com