node.js

lukas.weber@socialbakers.com

node.js

- javascript na serveru
- V8
- libuv = libev + libeio

Problém

Problém

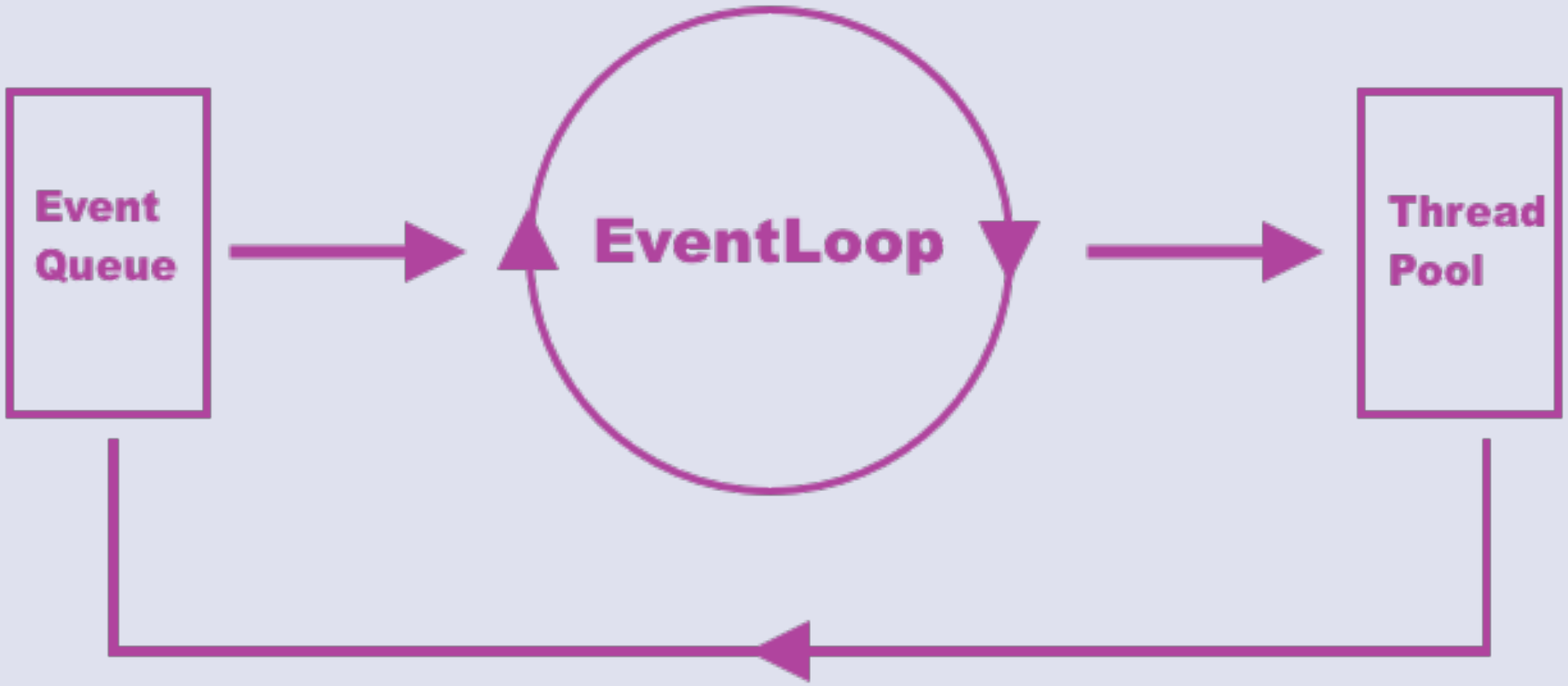
Řešení

- Více procesů - potřebuje load balancer, žere paměť, plánovač trpí
- fork() - pořád žere paměť, problémy při zabití rodiče (apache prefork)
- thready (vlákna) - nejlepší (apache worker), ale:
  - Problémové implementace - systémové vs. user apod.
  - Mutexy, synchronizace - těžké na vývoj
  - Reálná konkurence => WTF generátor

Event loop

- Jeden thread (EventLoop) pro váš kód
- Interní optimalizovaný threadpool mimo dosah programátora
- Událostní model

Event loop



nvm

- node version manager
- nvm install v0.10.20
- nvm use v0.10.20
- nvm alias default v0.10.20
- https://github.com/creationix/nvm

npm

- package manager
- http://npmsearch.com/
- package.json / node_modules
- module.exports.hello = function () {}
- module = require('./module');

package.json

```
{
    "name": "superduper",
    "version": "1.0.0",
    "description": "SuperDuper library"
    "homepage": "http://example.org/superduper",
    "author": { "name": "Loper Deve" },
    "repository": {
        "type": "git",
        "url": "git://github.com/xxx/superduper.git"
    },
    "engines": { "node": "~0.8.0" },
    "dependencies": {
        "express" : "~2.5.9",
        "socket.io" : "~0.9.6",
        "mustache" : "~0.4.0"
    }
}
```

Async programování

- Události vs. callbacky

- Opakující se situace => události

- Async předání výsledku => callback

EventEmitter

```
var events = require('events');
var util = require('util');

var Engine = function (type) {
    this.type = type;
    events.EventEmitter.call(this);
}

util.inherits(Engine, events.EventEmitter);

Engine.prototype.turnOn = function () {
      this.emit('engine_on');
}
Engine.prototype.turnOff = function () {
      this.emit('engine_off');
}
```

EventEmitter

```
var v8 = new Engine('v8');
v8.on('engine_on', function () {
      console.log(this.type + ' is on');
});
v8.on('engine_off', function () {
      console.log(this.type + ' is off');
});

v8.turnOn();
v8.turnOff();
```

Callbacky

```
fs.readFile('/my/file.txt', function (err, data) {
    if (err) {
        console.error(err);
        return;
    }

    console.log(data);
});
```

Callback hell

```
fs.readdir(source, function(err, files) {
  if (err) {
    console.log('Error finding files: ' + err)
  } else {
    files.forEach(function(filename, fileIndex) {
      console.log(filename)
      gm(source + filename).size(function(err, values) {
        if (err) {
          console.log('Error identifying file size: ' + err)
        } else {
          console.log(filename + ' : ' + values)
          aspect = (values.width / values.height)
          widths.forEach(function(width, widthIndex) {
            height = Math.round(width / aspect)
            console.log('resizing ' + filename + 'to ' + height + 'x' + height)
            this.resize(width, height).write(destination + 'w' + width + '_' + filename,
function(err) {
              if (err) console.log('Error writing file: ' + err)
            })
          }.bind(this))
        }
      })
    })
  }
})
```

async.js

```
async.parallel([
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even though
    // the second function had a shorter timeout.
});
```

async.js

```
async.parallel([
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even though
    // the second function had a shorter timeout.
});
```

async.js

```
async.parallel([
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even though
    // the second function had a shorter timeout.
});
```

async.js

```
async.parallel([
    function(callback){
        setTimeout(function(){
            callback(null, 'one');
        }, 200);
    },
    function(callback){
        setTimeout(function(){
            callback(null, 'two');
        }, 100);
    }
],
// optional callback
function(err, results){
    // the results array will equal ['one','two'] even though
    // the second function had a shorter timeout.
});
```

promises/futures/deferred

```
request(“http://localhost:8000")
      .then(function (responseText) {
            console.log(responseText)
})
.fail(function (err) {
      consoole.error(err);
});
```

express.js

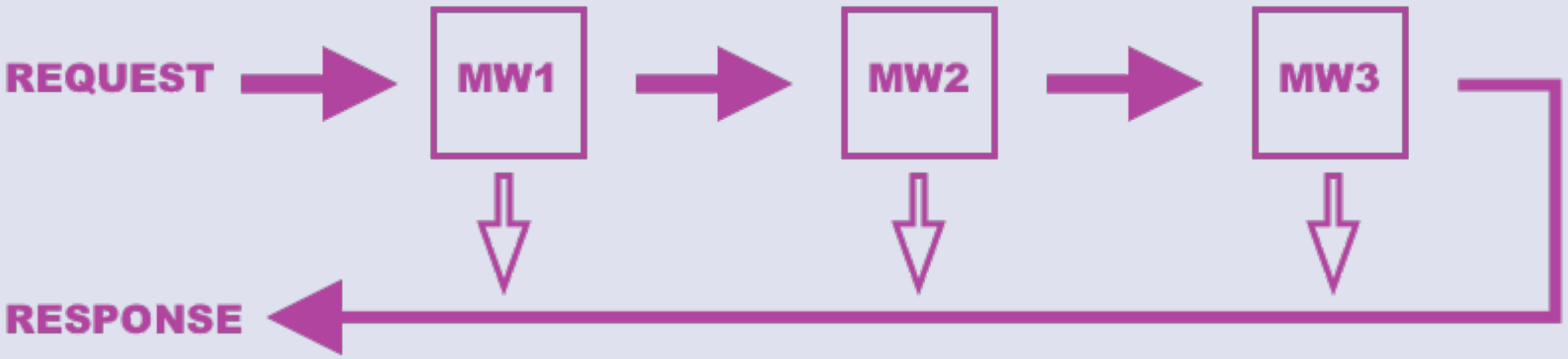
```
var express = require('express');

var app = express();

app.get('/hello', function (req, res) {
      res.send('world');
});

app.listen(8000);
```

middleware



nástroje

- nodemon - development, restartuje skript při změně
- node-inspector - debugger
- grunt.js - make, deploy
- pm2 - process monitor
- mocha/should - testovani
- esprima - ast
- jsdoc

next? >>>

- Streamy
- CoffeeScript/transpilery
- ES 6/Harmony - generátory, proxies, …

@techbakers

lukas.weber@socialbakers.com

twitter: @isnotgood