node.js kickstart

lukas.(benes|weber)@socialbakers.com

node.js ...why?

- incredibly fast
- better hardware utilization thanks to async I/O
- everyone can write javascript
- CommonJS

Event loop

- it's where your code is executed

- single thread

```
// not actual event loop, but close enough to
understand the concept

while (true) {
  handleFinishedIO();
  handleIncommingIO();
}

function handleIncomingIO(io) {
  doSomeIO();
}
```

a little bit of JavaScript

Closures

- closure means "function have reference to variables in place where has been declared, even when is passed deeper into stack"

```
var first = 'Hello';
var second = 'World!';

function welcome() {
  console.log(first + ' ' + second);
}

setTimeout(welcome, 2000); // execute in two seconds

first = 'Hell';
second = 'No!';

// will outpu 'Hell No!'
```

Closures

- retains references to variables so garbage collector can't kick in and remove unused variables

- we can take advantage of it e.g. pseudo private variables

```
var first = '', second = '';

var welcome = (function () {
  var first = 'Hello', second = 'World!';
  return function () {
    console.log(first + ' ' + second);
  }
})();

first = 'Hell', second = 'No!';

setTimeout(welcome, 2000);
// will properly output "Hello World!"
```

Callbacks

- handles result of asynchronous operation
- expected to be called only once
- by convention callback signature is:

```
function (err, result1, ...) { ... }
```

Callbacks

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

fs.readdir('./', function (err, files) {
   if (err) {
      console.error(err);
      return;
   }

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

Callback hell

- common name for too many nested callbacks (20 tabs to right...)
- can be solved by decomposition of dependent task
- async.js to the rescue!

Events

- reaction to event in system
- similar to callbacks (it's actually <u>is</u> callback)
- may be called multiple times
- doesn't have to follow callback signature
- best example of events use is node Stream API

Events

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

function Timer(tickInterval) {
   events.EventEmitter.call(this);
   var tickCnt = 0;
   var tick = (function () { this.emit('tick', ++tickCnt); }).bind(this);
   this.interval = setInterval(tick, tickInterval);
}

Timer.prototype = new events.EventEmitter;
Timer.prototype.stop = function () {
   clearInterval(this.interval);
}

var timer = new Timer(300);
timer.on('tick', function (tickNum) {
   console.log(tickNum);
   if (tickNum === 10) {
      timer.stop();
      console.log('Stop, Hammer time!');
   }
});
```

...promises

- They are combination of both
- Immediate value return in form of future/promise
- may and may not be fulfilled
- can be chained in various ways
- Q library, promise etc.
- Promises/A+ CommonJS proposal

...promises

```
var Promise = require('promise');

function tryLuck() {
   var future = new Promise(function (resolve, reject) {
      setTimeout(function () {
         if (Date.now() % 2) resolve('Luck!');
         else reject('Ha! Ha! Out of luck.')
      }, 1000);
   });

   return future;
}

var result = tryLuck();
result.then(
   function (msg) { console.log('just received: ' + msg); },
   function (err) { console.log('error: ' + err); }
);
```

Callbacks vs. Events

- Use events only when something is happening "itself"
  - client request is initiated by user -> handle event
  - you're performing request -> use callback
  - sometimes it's make sense to use both, then stick with callbacks

Callbacks vs. Events

```
var auth = new AuthService;

// don't do this
auth.login('user', 'password');
auth.on('login', function (user) {
  // handle `user`
});

// instead do:
auth.login('user', 'password', function (err, user) {
  if (err) {
    // handle `err`
    return
  }
  // handle `user`
});
```

node.js

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

modules

- module is just ordinary .js file
- evaluated on require() call in private scope
- require() returns module.exports of sourced file

modules

- module_a.js

```
var b = require('./module_b');
b.sayHello();
```

- module_b.js

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

module.exports.sayHello = function () {
  console.log('Hello you on ' + os.hostname());
}

if (!module.parent) {
  console.log('running module_b directly');
}
```

npm

- package manager
- http://npmsearch.com/
- package.json / node_modules
- C++ packages not always compatible with all OSes
- SemVer module versioning 2.1.1 (major.minor.patch)

express.js

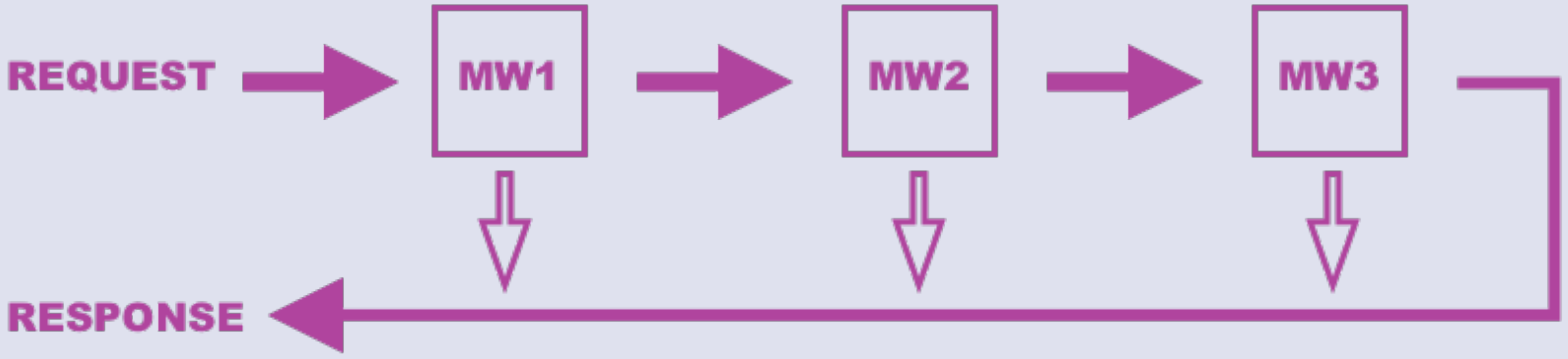
```
var express = require('express');

var app = express();

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

app.listen(8000, function () {
  var ip = this.address();
  console.log('Listening on '
    + ip.address + ':' + ip.port);
});
```

middleware



tools to look at

- http://nodemon.io/
- http://gruntjs.com/
- https://github.com/Unitech/pm2
- http://visionmedia.github.io/mocha/
- http://esprima.org/
- http://usejsdoc.org/

tools to look at

- https://github.com/node-inspector/node-inspector
- https://github.com/caolan/async
- https://github.com/mikeal/request
- http://underscorejs.org/
- http://semver.org/
- follow on Twitter @tjholowaychuk, @izs, @creationix, @StrongLoop

@techbakers

lukas.(benes|weber)@socialbakers.com