



Node.js from zero to hero

Nicola Del Gobbo – Mauro Doganieri

WHAT IS Node.js?

Node.js is a **JavaScript runtime** built on Chrome's **V8 JavaScript engine**

Node.js uses an event-driven, **non-blocking I/O** model that makes it lightweight and efficient

Node.js' package ecosystem, **npm**, is the largest ecosystem of open source libraries in the world



A **JavaScript Runtime**



Asynchronous I/O based on **event loops**

~100k LOC of JS and C++

Node **glue**



Ecosystem of **packages**

WHAT CAN I DO WITH **Node.js**?

EVERYTHING

Great thank you to Node.js **Community** and **Working Groups**



Packages

603.284

Downloads · Last Day

201.263.400

Downloads · Last Week

3.469.805.066

WHAT CAN I DO WITH Node.js?

express



CLI APPLICATION

DISRTRUBUTED SYSTEM

REAL TIME SYSTEM

DESKTOP APPLICATION

ARCHITECTURE

Node.js API

Node.js Bindings

C / C++ Addons



c-ares

HTTP
parser

Open
SSL

zlib

nghttp2

CONCURRENCY MODEL?

ASYNCHRONOUS I/O

SINGLE THREADED

SYNCHRONOUS MODEL?

```
var result = db.query('select ...');  
// use result
```

Blocking the whole process or you need to have **multiple execution stacks**

ASYNCHRONOUS MODEL?

```
db.query('select ...', function (result) {  
  ... // use result  
})
```

```
// Execution continue  
console.log('Do other stuff ...')
```

The **main** process is **never blocked**. No strategy is required to handle competing requests

EVENT LOOP



ASYNCHRONOUS IN LOW LEVEL

```
'use strict'
```

```
const gs = require('ghostscript4js')
```

```
let cmd = '-sDEVICE=pngalpha -o my.png -sDEVICE=pngalpha -r144 my.pdf'
gs.execute(cmd, function (err) {
  if (err) {
    console.log("Ooops... something wrong happened")
  }
})
```

```

class GhostscriptWorker : public AsyncWorker
{
public:
    GhostscriptWorker(Callback *callback, string RAWcmd)
        : AsyncWorker(callback), RAWcmd(RAWcmd), res(0) {}
    ~GhostscriptWorker() {}

```

```

void Execute()
{
    res = 0;
    vector<string> explodedCmd;
    istream iss(RAWcmd);
    for (string RAWcmd; iss >> RAWcmd;)
        explodedCmd.push_back(RAWcmd);
    int gsargc = static_cast<int>(explodedCmd.size());
    char **gsargv = new char *[gsargc];
    for (int i = 0; i < gsargc; i++)
    {
        gsargv[i] = (char *)explodedCmd[i].c_str();
    }
    try
    {
        GhostscriptManager *gm = GhostscriptManager::GetInstance();
        gm->Execute(gsargc, gsargv);
        delete[] gsargv;
        res = 0;
    }
    catch (exception &e)
    {
        delete[] gsargv;
        msg << e.what();
        res = 1;
    }
}

```

This code will be executed on the Worker Thread

```

void HandleOKCallback()
{
    Nan::HandleScope();
    Local<Value> argv[1];
    if (res == 0)
    {
        argv[0] = Null();
    }
    else
    {
        argv[0] = Error(Nan::New<String>(msg.str()).ToLocalChecked());
    }
    callback->Call(1, argv);
}

```

When execute method end its computation the results will be reassembled and passed to JS context

```

private:
    string RAWcmd;
    int res;
    stringstream msg;

```

```

};

```

NAN_METHOD(Execute)

```

{
    Callback *callback = new Callback(info[1].As<Function>());
    Local<String> JScmd = Local<String>::Cast(info[0]);
    AsyncQueueWorker(new GhostscriptWorker(callback, *String::Utf8Value(JScmd)));
    info.GetReturnValue().SetUndefined();
}

```

Return immediately

MODULARITY

CommonJS Module

ES6 Module from v8.5.0

“Se non hai provato Node.js non sai cos'è la
modularità e il riuso”

Matteo Collina

MODULARITY

```
'use strict'
```

```
function Person (opts) {  
  ...this._firstName = opts.firstName || ''  
  ...this._lastName = opts.lastName || ''  
}  
  
Person.prototype.toString = function toString() {  
  ...return `First name: ${this._firstName}  
  ...Last name: ${this._lastName}`  
}
```

```
module.exports = Person
```

```
'use strict'
```

```
function isPrime(p) {  
  ...const upper = Math.sqrt(p)  
  ...for(let i = 2; i <= upper; i++) {  
    ...if (p % i === 0) {  
      ...return false  
    }  
  }  
  ...return true  
}
```

```
exports.isPrime = isPrime
```

MODULARITY

```
'use strict'
```

```
const Person = require('./Person')
```

```
const p = new Person({  
  ...firstName: 'Nicola',  
  ...lastName: 'Del Gobbo'  
})
```

```
console.log(p.toString())
```

```
'use strict'
```

```
const isPrime = require('./isPrime')
```

```
console.log(isPrime(3))
```

EVENT EMITTER

EventEmitter is a very important class in Node.js. It provides a **channel for events to be dispatched** and **listeners notified**. Many objects you will encounter in Node.js inherit from EventEmitter

EVENT EMITTER

```
'use strict'

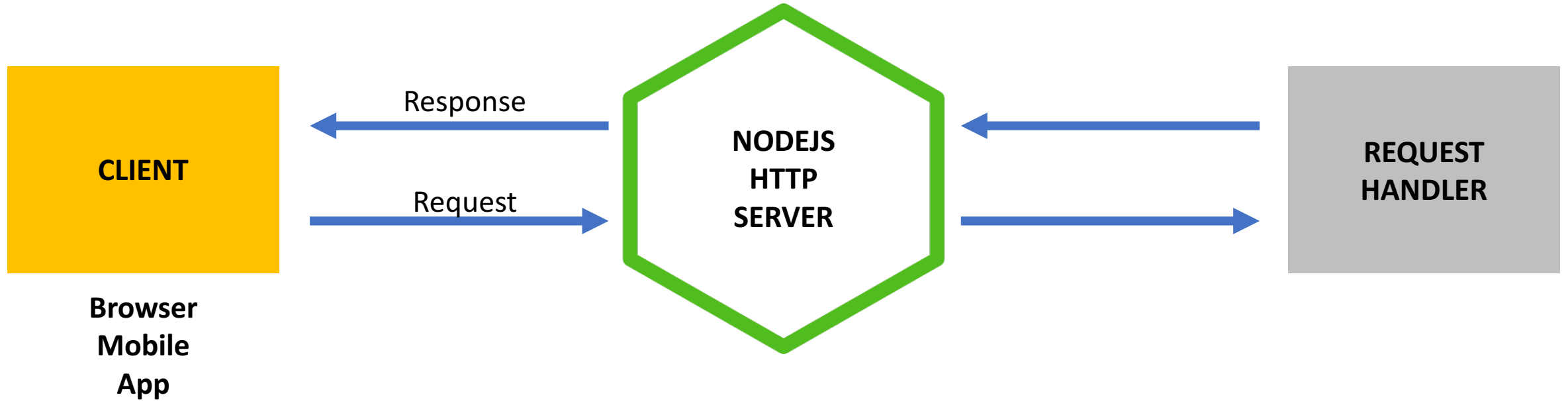
const EventEmitter = require('events').EventEmitter

class User extends EventEmitter {
  ...constructor () {
    ...// do some initialization tasks
    ...}

  ...addUser(user) {
    ...// register user on db
    ...// emit event
    ...this.emit('user:added', {userName: 'NickNaso', password: 'nodejs'})
    ...}
}

User.on('user:added', (data) => {
  ...// do something with data
})
```


WEB APPLICATION



<http://your-web-application>

~40k req/s

Hard to maintain

```
'use strict'
```

```
const http = require("http")
```

Import http module

```
const url = require("url")
```

```
function onRequest(request, response) {
```

Listener for the incoming request

```
  let pathname = url.parse(request.url).pathname  
  console.log("Request for " + pathname + " received.")
```

Route handler

```
  if (pathname === "/start") {  
    response.writeHead(200, { "Content-Type": "text/plain" })  
    response.write("Hello")  
    response.end()
```

```
  } else if (pathname === "/finish") {  
    response.writeHead(200, { "Content-Type": "text/plain" })  
    response.write("Goodbye")  
    response.end()
```

```
  } else {  
    response.writeHead(404, { "Content-Type": "text/plain" })  
    response.end("404 Not Found")
```

```
  }  
}
```

```
http.createServer(onRequest).listen(5000)
```

```
console.log("Server has started.")
```


FRAMEWORKS



express

EXPRESS

 [expressjs](#) / [express](#)

 Unwatch ▾


1,736


★ Unstar


34,685

 Fork

6,261


 Code

 Issues 104

 Pull requests 42

 Projects 0

 Wiki

 Insights

Fast, unopinionated, minimalist web framework for node. <https://expressjs.com>

javascript

nodejs

express

server



Stats

788.780 downloads in the last day

4.310.178 downloads in the last week

18.740.809 downloads in the last month

EXPRESS

- Minimalist
- Unopinionated
- Fast (about **21k** req/sec)
- Simple (**do one thing well** philosophy from Unix world)
- Wrapper of **http** core module

```
'use strict'
```

```
const http = require("http")
```

```
const express = require('express')
```

Import and create

```
const app = express()
```

```
app.get('/start', (req, res) => {  
  res.status(200).send('Hello')  
})
```

Route handler

```
app.get('/finish', (re, res) => {  
  res.status(200).send('Goodbye')  
})
```

```
http.createServer(app).listen(5000)
```

```
console.log("Server has started.")
```

For web sockets bind express to http module

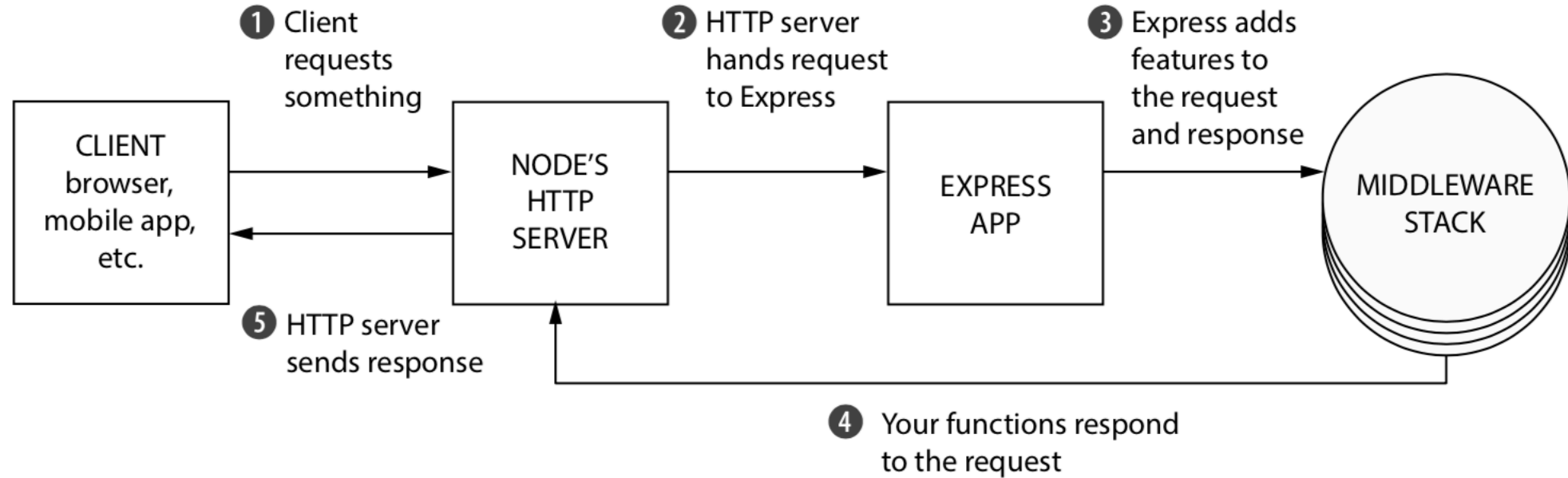
EXPRESS MILESTONES

- Middleware
- Error handler
- Router
- Views / template engine

MIDDLEWARES

It's always a question to manipulate the **Request**
and **Response** object

MIDDLEWARES



MIDDLEWARES

```
'use strict'
```

```
const path = require('path')
const http = require('http')
const express = require('express')
const morgan = require('morgan')
const serve = require('express').static
```

```
const app = express()
```

Attach middleware to Express

```
app.use(morgan('combined'))
app.use(serve(path.join(__dirname, 'public')))
```

```
app.get('/', (req, res) => {
  res.status(200).sendFile('/index.html')
})
```

```
http.createServer(app).listen(5000)
console.log("Server has started.")
```

MIDDLEWARES

```
'use strict'
```

```
const bannedIps = ['192.168.0.1', '192.168.0.2', '192.168.0.3']
```

```
function myMiddleware (req, res, next) {
```

```
  // ... DO SOMETHING WITH REQUEST AND RESPONSE
```

```
  if (bannedIps.includes(req.ip)) {
```

```
    const err = new Error('Your IP is banned go away')
```

```
    next(err)
```

```
  } else {
```

```
    next()
```

```
  }
```

```
}
```

```
'use strict'
```

```
function myMiddleware (bannedIps) {
```

```
  // ... DO SOMETHING WITH BANNED IPS
```

```
  return function (req, res, next) {
```

```
    // ... DO SOMETHING WITH REQUEST AND RESPONSE
```

```
    if (bannedIps.includes(req.ip)) {
```

```
      const err = new Error('Your IP is banned go away')
```

```
      next(err)
```

```
    } else {
```

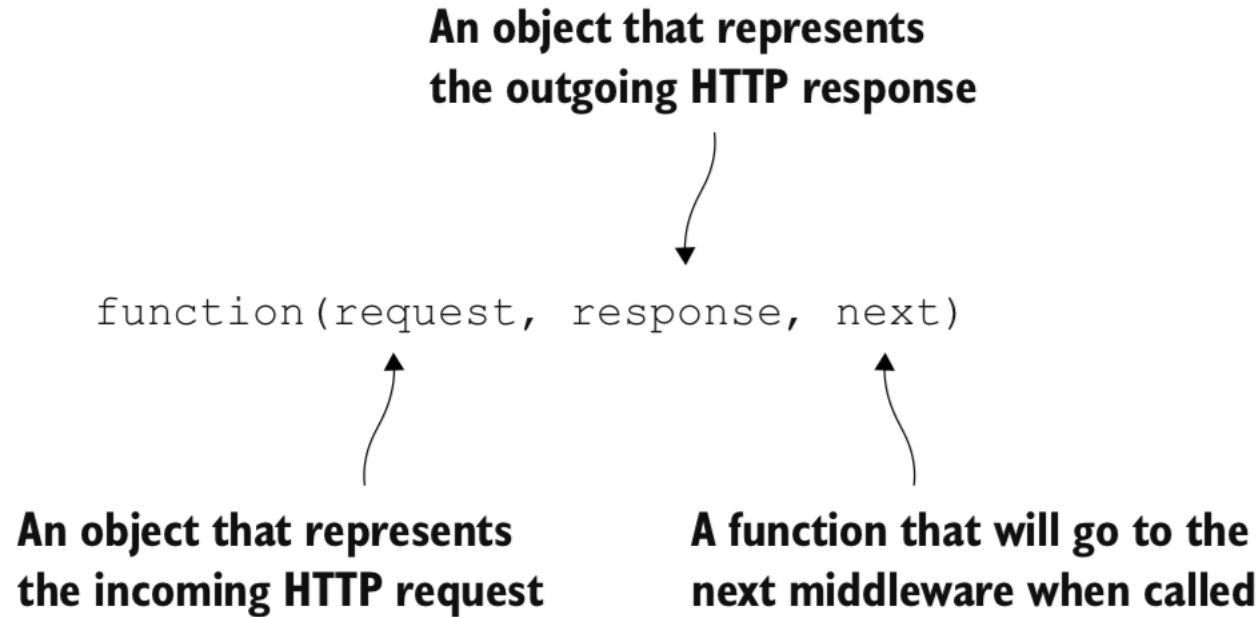
```
      next()
```

```
    }
```

```
  }
```

```
}
```

MIDDLEWARES

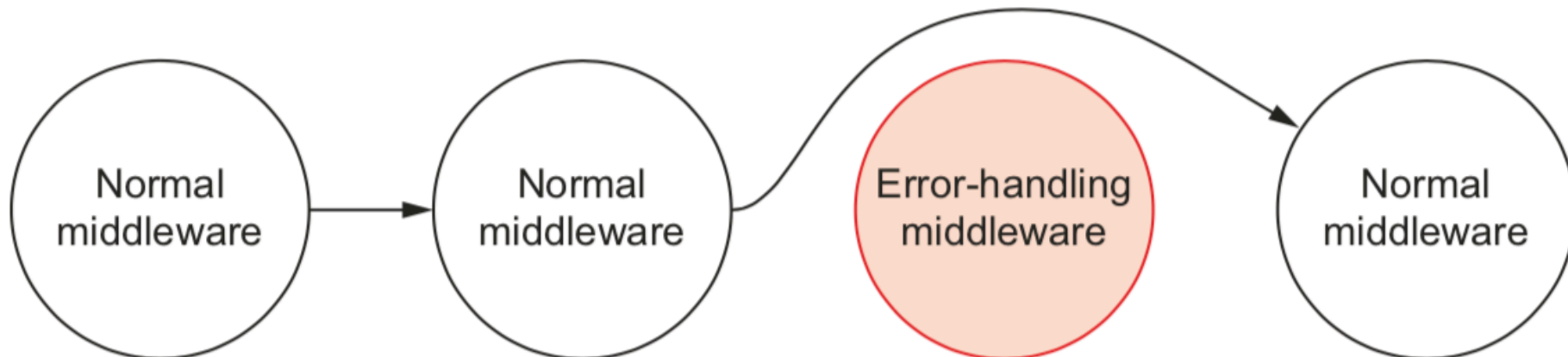


<http://expressjs.com/en/resources/middleware.html>

ERROR HANDLER

```
'use strict'
```

```
function errorHandler () {  
  return function (err, req, res, next) {  
    // ... PARSE YOUR ERROR  
    // ... DO SOMETHING WITH REQUEST AND RESPONSE  
    // ... IDENTIFY STATUS CODE AND MESSAGE FOR YOUR RESPONSE  
  }  
}
```



ROUTING

```
app.get('/songs', (req, res, next) => {  
  // ... DO SOMETHING ON YOUR ROUTE  
})  
  
app.get('/songs/:title', (req, res, next) => {  
  // ... DO SOMETHING ON YOUR ROUTE  
})  
  
app.post('/songs', (req, res, next) => {  
  // ... DO SOMETHING ON YOUR ROUTE  
})  
  
app.put('/songs/:title', (req, res, next) => {  
  // ... DO SOMETHING ON YOUR ROUTE  
})  
  
app.delete('/songs/:title', (req, res, next) => {  
  // ... DO SOMETHING ON YOUR ROUTE  
})
```

Routing refers to determining how an application responds to a client request to a particular endpoint, which is a URI (or path) and a specific HTTP request method

VALIDATE YOUR INPUT

Ajv

The fastest JSON Schema validator for
Node.js and browser

npm install ajv

<https://github.com/epoberezkin/ajv>

VIEWS / TEMPLATE ENGINE

Pug - Mustache - Dust - Nunjucks - EJS

```
app.set('view engine', 'ejs') Set the engine
```

```
app.engine('ejs', require('ejs').__express) Register the engine
```

```
app.set('views', path.join(__dirname, 'views')) Set views folder
```


SECURITY

Helmet helps you secure your Express apps by setting various HTTP headers

npm install helmet

<https://github.com/helmetjs/helmet>

LOGGER

Log everything that happens in your application

Pay attention there is a **cost** for logging

npm install winston

<https://github.com/winstonjs/winston>

DOMANDE?



GRAZIE

<https://github.com/NickNaso/nodejs-tsw-2017>