HTTP, Node.js and Koa

Feng Zhou

2016-12-02

Learn you some cli for great good

get Gow

- Gnu on windows
- `https://github.com/bmatzelle/gow`

where am i

pwd print working directory

ls list directory contents

move

cd change directory

file inspection

file print file type

cat concat and print

create

mkdir make directory

touch can be used to create an empty file

copy and rename

cp copy

mv move

destroy

rm remove

rmdir safely remove a directory

HTTP 101

TCP Socket

- telephone
- bi-directional
- ip address and port

Port

- physical port like usb port
- 2^{16}
- port for http server is 80
- bind to port lower than 1024 requires root privilage

HTTP

HTTP Hypertext Transfer Protocol

HTML ?

Request

```
GET /user/1 HTTP/1.1
Host: localhost:3000
User-Agent: curl/7.47.0
Accept: */*
```

View Request with curl

```
curl -vs bing.com 2>&1 | grep '^>'
```

Request

- method
- path
- headers

```
GET /user/1 HTTP/1.1
Host: localhost:3000
User-Agent: curl/7.47.0
Accept: */*
```

Methods

- GET
- POST
- PUT
- DELETE
- HEAD
- PATCH

Response

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Content-Length: 11
Date: Wed, 26 Oct 2016 05:29:14 GMT
Connection: keep-alive

hello there
```

View Response with curl

```
curl -i gnu.com
```

Response

- status code
- headers
- body

```
HTTP/1.1 200 OK
Content-Type: text/plain; charset=utf-8
Content-Length: 11
Date: Wed, 26 Oct 2016 05:29:14 GMT
Connection: keep-alive

hello there
```

Status Codes

200 OK

301 Moved Permanently

304 Not Modified

400 Bad Request

401 Unauthorized

403 Forbidden

404 Not Found

500 Internal Server Error

Status Codes

```
curl -i bing.com
```

```
curl -i cn.bing.com/.png
```

Request and response

- stateless

Tracking users

- cookie
- referer
- variables embedded in url

Cookies

- set by server
- carried in the request header by client(browser)

Cookies

```
curl bing.com -siL | grep Set-Cookie
```

Cookies

- authentication
- tracking client
- signed/ecrypted/httponly/expiration

Websocket

- bidirectional
- realtime

HTTP/2

- binary protocol
- better speed
  - multiple request, one TCP connection
  - header compression

tcpdump

- `http://www.tcpdump.org/`
- state-of-the-art traffic inspecting tool
- not very friendly for newbies

```
sudo tcpdump -X -nni lo port 3000 and tcp
```

tcpflow

- `https://github.com/simsong/tcpflow`
- output is easier to read for http inspection

```
sudo tcpflow -X /dev/null -i any -C port 3000
```

gor

- https://goreplay.org/
- cross platform support
- easy to install(single binary file, written in golang)

```
sudo gor --input-raw-track-response \
  --input-raw :3000 --output-stdout
```

mitmproxy

- `https://mitmproxy.org/`
- has a gui
- designed for http
- available as a python module(but not available on windows)

```
export http_proxy=http://127.0.0.1:8080
curl localhost:3000
```

JavaScript

JavaScript

Any application that can be written in JavaScript, will eventually be written in JavaScript.

The Origin of JavaScript

- 1995
- Nestscape
- Brendan Eich
- 10 days

ECMAScript

ECMAScript the standard

JavaScript the implemantation of ECMAScript

Transpiling

Transpiling source code to source code compilation

```
ES 6 -> ES 5
CoffeeScript -> JavaScript
TypeScript -> JavaScript
Elm -> JavaScript
```

What is Babel

- Babel compiles newer version js to older version js
- `https://babeljs.io/`

npm and yarn

npm nodejs package manager

yarn facebook's improved package manger

get yarn

```
npm i -g yarn
```

or

```
npm install --global yarn
```

Using Babel

```
yarn global add babel-cli
```

this is not the recommended way

Babel plugins

```
yarn add babel-plugin-transform-async-to-generator
yarn add babel-plugin-transform-es2015-modules-commonjs
```

Babel plugins

add to package.json

```
{
  "babel": {
    "plugins": [
      "transform-async-to-generator",
      "transform-es2015-modules-commonjs"
    ]
  }
}
```

Babel plugins

or add to .babelrc

```
{
  "plugins": [
    "transform-async-to-generator",
    "transform-es2015-modules-commonjs"
  ]
}
```

Using Babel

```
cat 00-babel.js
babel 00-babel.js
```

Using Babel REPL

```
babel-node
```

Functions

- common functions
- generator functions
- async functions

Common Functions

```
function add(num, num2) {
  return num + num2
}

let add = function(num, num2) {
    return num + num2
}
```

Generator Functions

- function can only return one value
- generators can yield multiple values

Generator Functions

```
function* range(start, end) {
  while (start <= end) {
    yield start
    start = start + 1
  }
}

for (let i of range(1, 9)) {
  console.log(i)
}
```

Async Functions

async functions alway return a Promise

```
truth = async function() {
  return true
}

truth()
```

Async Functions

```
async function() {
  await fetch()
}
```

Fat Arrow

- no context(this)
- terse

```
let add = (num, num2) => num + num2

let add = (num, num2) => {
  return num + num2
}
```

String interpolation

```
sum = 2
`sum: ${sum}`
```

String interpolation

```
`sum: ${1 + 1}`
```

JSON

JSON JavaScript Object Notation, a subset of JavaScript

```
{
  "key": "value",
  "object": {
    "number": 1.2,
  },
  "array": [null, {"nested": true}]
}
```

JSON

- less verbose than XML
- very popular format for data exechange

JSON

encoding

```
JSON.strigify(myObject)
```

decoding

```
JSON.parse(inputString)
```

Koa

The Origin

Successor of Express, the most popular nodejs web framework

Compared with Express

- both are middleware based
- express uses callbacks
- koa uses async function or generator function
- koa is slimer, no middleware bundled

Example Koa

```
import Koa from "koa"
let app = new Koa()

app.use(async (ctx) => {
  ctx.body = await Promise.resolve("koa")
})

app.listen(3000)
```

Example Express

```
import Express from "express"
let app = Express()

app.get("/", (req, res) => {
  Promise.resolve("express").then(body => {
    res.send(body)
  })
})

app.listen(3000)
```

Callback Hell

or Pyramid of Doom

```
operation1((err, result1) => {
  operation2((err, result2) => {
    operation3((err, result3) => {
      ...
    })
  })
})
```

Callback Hell

```
async () => {
  result1 = await operation1()
  result2 = await operation2()
  result3 = await operation3()
}
```

Middleware

A logging middleware

```
import Koa from "koa"
let app = new Koa()

app.use(async (ctx, next) => {
    console.log(`${ctx.method} ${ctx.path}`)
    await next()
})

app.use(ctx => { ctx.body = "hello there" })

app.listen(3000)
```

A logging middleware

```
curl localhost:3000
curl -X POST localhost:3000
curl -X DELETE localhost:3000/users/1
```

A timing middleware

```
let timeRequest = async (ctx, next) => {
    let requestStarted = new Date()
    await next()
    console.log(`took: ${new Date() - requestStarted} ms`)
}

app.use(timeRequest)
```

Middleware is like Onion

```
import Koa from "koa"
let app = new Koa()

app.use(async (ctx, next) => {
  console.log('>> one')
  await next()
  console.log('<< one')
})
```

Middleware is like Onion

```
app.use(async (ctx, next) => {
  console.log('>> two')
  ctx.body = 'two'
  await next()
  console.log('<< two')
});

app.use(async (ctx, next) => {
  console.log('>> three')
  await next()
  console.log('<< three')
})

app.listen(3000)
```

Passing options to middlewares

```
let myMiddleware = (options) => async (ctx, next) => {
  await next()
}
```

Mounting a middleware

```
app.use(middleware(opts))
```

Semantic versioning

semver for short, major.minor.patch, 1.20.1

- major for incompatable change
- minor for new feature
- patch for bug fix
- first table release should be 1.0.0

Npm and semver

```
yarn add koa@next
```

Routing

The process of mapping requests to handlers

- method
- path

Using koa-router

koa has no built-in routing support

```
yarn add koa-router@next
```

Using koa-router

```
import Router from "koa-router"
let router = new Router()

router.get('/', ctx => {
  ctx.body = 'router'
})

app.use(router.routes())
```

Extracting params from url

```
router.get('/hello/:name', ctx => {
  ctx.body = `hello ${ctx.params.name}`
})
```

More HTTP methods

```
router.post('/', ctx => {
  ctx.body = 'it was POST'
})
```

Popular engines

pug	formerly know as jade
mustache	minimal, logic less, language-agnostic
nunjucks	inspired by jinja2
ejs	old-style

Koa views

```
yarn add koa-views@next ejs
```

Configure view engine

```
import views from 'koa-views'

app.use(views(`${__dirname}/views`, {
  map: {html: 'ejs'}
}))
```

Passing variables

in handler

```
ctx.render("index", {key: val})
```

in middleware

```
ctx.state.name = value
```

koa-static

```
yarn add koa-static@next

import serve from "koa-static"

app.use(serve(`${__dirname}/public`))
```

form

```
<form method="POST">
  <div>
    <input name="username">
  </div>
  <div>
    <input type="password" name="password">
  </div>
  <input type="submit" value="Submit">
</form>
```

body parser

- https://github.com/koajs/bodyparser

```
yarn add koa-bodyparser@next
```

body parser

```
import bodyParser from 'koa-bodyparser'

app.use(bodyParser())

router.post('/', ctx => {
  console.log(ctx.request.body)
  ctx.body = ctx.request.body
})
```