# Get Started with Koa

Feng Zhou

2016-10-21

## HTTP 101

## 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

## Request

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

## Request

- method
- path
- headers

## 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
```

## Response

- status code
- headers
- body

## Status codes

|  |  |
|---|---|
| **200** | OK |
| **304** | Not Modified |
| **400** | Bad Request |
| **401** | Unauthorized |
| **403** | Forbidden |
| **404** | Not Found |

## Request and response

- stateless

## Cookies

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

## Cookies

- authentication
- tracking client
- signed/ecrypted

## Websocket

- realtime

## HTTP/2

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

## tcpdump

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

## tcpflow

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

## gor

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

## mitmproxy

```
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
- ten days

## ECMAScript

**ECMAScript** the standard

**JavaScript** the implemantation of ECMAScript

## Transpiling

**Transpling** 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/`

## Using Babel

```
npm install --global babel-cli
```

this is not the recommended way!

## Using Babel

```
$ babel script.js
```

## Using Babel

```
$ babel-node
```

## Configure babel plugins

add to `package.json`

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

## Functions

- common functions
- generator functions
- async functions

## Common Functions

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

const 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

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

## Fat Arrow

- no context(this)
- terse

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

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

## 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"
const app = new Koa

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

app.listen(3000)
```

## Example Express

```
import Express from "express"
const app = new Express

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

app.listen(3000)
```

## Callback Hell

or Pyramid of Doom

## Middleware

## A logging middleware

```
import Koa from "koa"
const 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)

curl localhost:3000
```

## A timing middleware

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

app.use(timeRequest)
```

## Middleware is like Onion

```
import Koa from "koa"
const 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

```
const myMiddleware = (options) => async (ctx, 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

```
npm install koa@next
```

## Routing

The process of mapping requests to handlers

- method
- path

## Using koa-router

koa has no built-in routing support

```
npm install --save koa-router@next
```

## Using koa-router

```
import Router from "koa-router"
const 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

```
npm install --save 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

```
npm install --save koa-static@next

import serve from "koa-static"

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