

Javascript on the Server

Thierry Sans

Node.js

- Runs on Chrome V8 Javascript engine
- Non blocking-I/O (a.k.a asynchronous, a. k.a event-driven)
- No restrictions (unlike when js is running on the browser)

Example

src/node/readfile.js

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

fs.readFile('helloworld.txt', 'utf8', function(err, data) {
  if (err) console.log(err);
  return console.log("output 1");
});

console.log("output 2");
```

console

```
$ node example.js
output 2
output 1
```

Building an HTTP server with Node.js

src/node/httpserver.js

```
const http = require('http');
const PORT = 3000;

var handler = function(req, res){
  console.log("Method:", req.method);
  console.log("Url:", req.url);
  console.log("Headers:", req.headers);
  res.end('hello world!');
};

http.createServer(handler).listen(PORT, function (err) {
  if (err) console.log(err);
  else console.log("HTTP server on http://localhost:%s", PORT);
});
```

Routing HTTP requests

Process HTTP requests and execute different actions based on

- the request method
 - the url path
 - whether the user is authenticated
 - ect ...
- ⦿ A router can be written from scratch (but it is tedious)
 - ⦿ Use the backend framework **Express.js**

Express.js - HTTP Methods

src/express-examples/01_httpmethods.js

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

// curl localhost:3000/
app.get('/', function (req, res, next) {
  res.end("Hello Get!");
});

// curl -X POST localhost:3000/
app.post('/', function (req, res, next) {
  res.end("Hello Post!");
});

const http = require('http');
const PORT = 3000;

http.createServer(app).listen(PORT, function (err) {
  if (err) console.log(err);
  else console.log("HTTP server on http://localhost:%s", PORT);
});
```

Express.js - Routing based on the path

src/express-examples/02_routing.js

```
// curl localhost:3000/  
app.get('/', function (req, res, next) {  
    res.end(req.path + ": the root");  
});  
  
// curl localhost:3000/messages/  
app.get('/messages/', function (req, res, next) {  
    res.end(req.path + ": get all messages");  
});  
  
// curl localhost:3000/messages/1234/  
app.get('/messages/:id/', function (req, res, next) {  
    res.end(req.path + ": get the message " + req.params.id);  
});
```

Express.js - body encoding

The body of HTTP request and response is a string

➔ **Problem:** how to send data structure between the frontend and backend?

➔ **Solution:** encode them either using:

- ✓ URI encoding (sometimes used)

see `src/express-examples/04_body-uri-encoded.js`

- ✓ XML encoding (rarely used these days)

- ✓ JSON encoding (very frequently used these days)

see `src/express-examples/05_body-json-encoded.js`