

COMP 431/531: Web Development

Lecture 16: Backend - Express

Mack Joyner (mjoyner@rice.edu)

<https://www.clear.rice.edu/comp431>



Back End Development

PART II
**Web Servers
Backend
Architecture
Testing
Web Hosting
Databases**



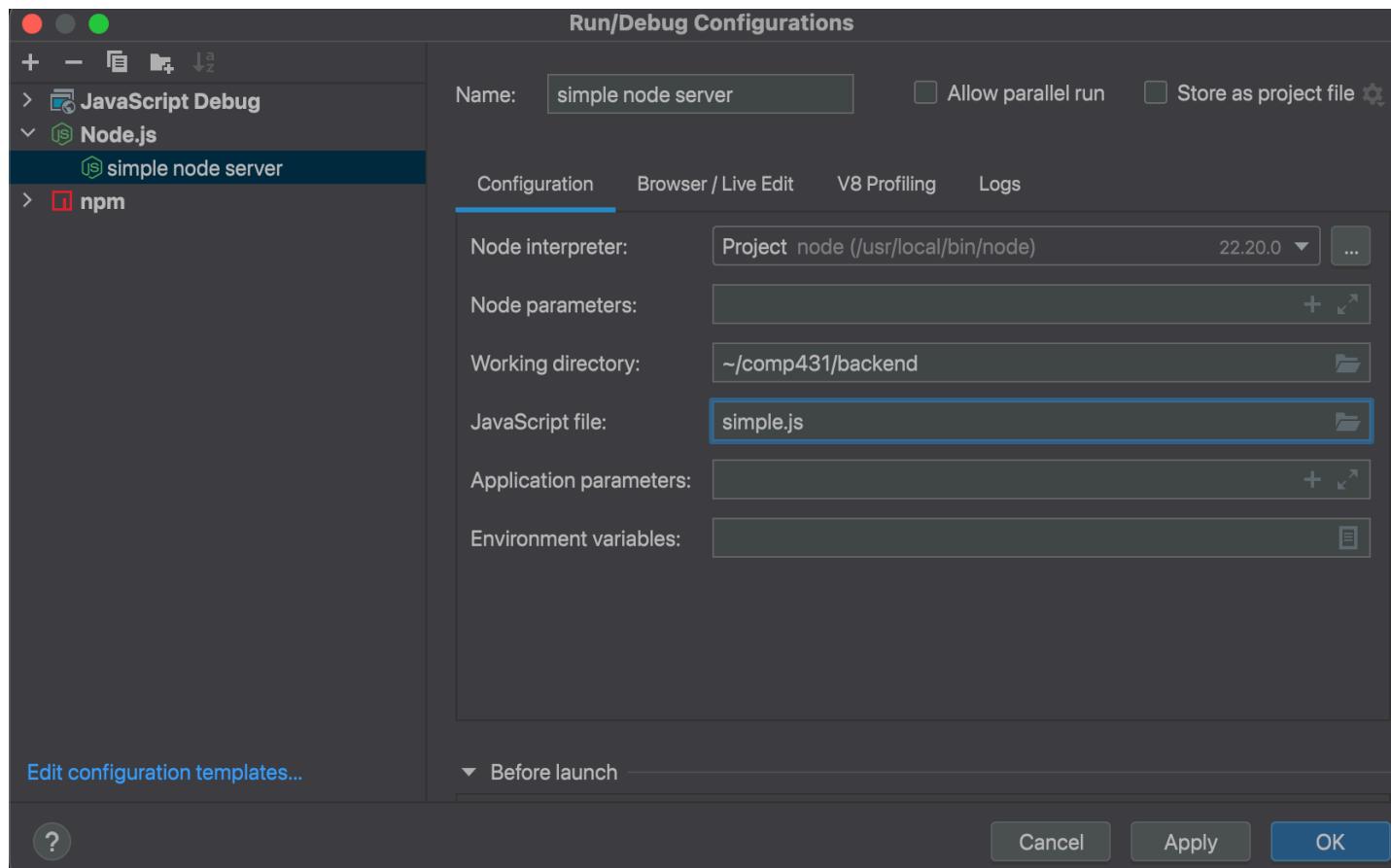
Announcements & Reminders

HW #5 (Front-end) due Thu. Nov. 6th at 11:59pm

classroom hw5 repo: <https://classroom.github.com/a/hlFk8cqA>



Debugging Node.js



<https://www.jetbrains.com/help/idea/running-and-debugging-node-js.html>

<https://plugins.jetbrains.com/plugin/6098-nodejs>



Talend API Tester

The screenshot shows the Talend API Tester interface. At the top, there are tabs for 'API Tester' (selected), 'Requests', and 'Scenarios'. Below the tabs, a search bar contains 'http://localhost:3000/articles' with a dropdown menu showing 'GET'. To the right of the search bar is a 'Send' button. Underneath the search bar, there are sections for 'QUERY PARAMETERS', 'HEADERS', and 'BODY'. The 'BODY' section notes that 'XHR does not allow payloads for GET request.' In the 'Response' section, the status is '200 OK'. The 'HEADERS' panel shows standard HTTP headers like X-Powered-By, Content-Type, ETag, Date, and Connection. The 'BODY' panel displays a JSON array of two posts:

```
[{"id": 0, "author": "Mack", "body": "Post 1"}, {"id": 1, "author": "Jack", "body": "Post 2"}]
```

Add as a Chrome extension from [Chrome web store](#)



Talend API Tester

METHOD SCHEME // HOST [":" PORT] [PATH ["?" QUERY]]
http://localhost:3000/article
length: 29 byte(s)

QUERY PARAMETERS

HEADERS
 Content-Type : application/json
+ Add header

BODY | | Enable body evaluation
length: 54 bytes

```
1 {  
2   "author": "Mack",  
3   "body": "Here is a new post"  
4 }
```

200 OK

HEADERS BODY
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 177 bytes
ETag: W/"b1-luiPoDCSLL7sutIFxWvTC56KH9E"
Date: Wed, 04 Oct 2023 22:05:14 GM
T
Connection: keep-alive
Keep-Alive: timeout=5

COMPLETE REQUEST HEADERS

```
[  
  {  
    id: 0,  
    author: "Mack",  
    body: "Post 1"  
  },  
  {  
    id: 1,  
    author: "Jack",  
    body: "Post 2"  
  },  
  {  
    id: 2,  
    author: "Zack",  
    body: "Post 3"  
  },  
  {  
    id: 3,  
    author: "Mack",  
    body: "Here is a new post"  
  }]
```



Talend API Tester Demo



Preprocess Request Body

- Must preprocess req body
 - chunk data
- Would be better to have someone do that for us
 - can use middleware

```
1  var http = require('http');
2
3  var host = '127.0.0.1';
4  var port = 3333;
5
6  http.createServer(preprocess).listen(port, host);
7  console.log('Server running at http://'+host+':'+port);
8
9  function preprocess(req, res) {
10    var body = '';
11    req.on('data', function(chunk) {
12      body += chunk
13    });
14    req.on('end', function() {
15      req.body = body;
16      server(req, res)
17    })
18  }
```



Middleware

- Middleware is anything you put between the server and the final application/framework
- Middleware is compliant
 - accept a request/response and pass it along
- Can modify the request/response
 - check for authentication
 - add(strip headers
 - format or transform content





Express

- Express is part of the MEAN or MERN stack
 - Mongo-Express-Angular-Node
 - Mongo-Express-React-Node
- Express adds to Node a number of helpful libraries
 - minimalist approach
 - middleware

Express

Fast, unopinionated,
minimalist web framework for
Node.js



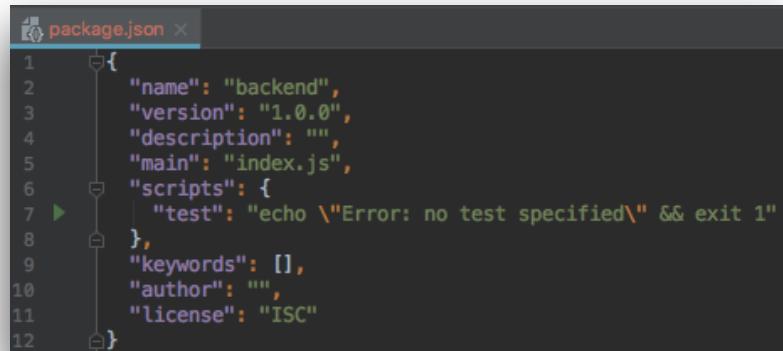
Installing Express

```
>> mkdir backend
```

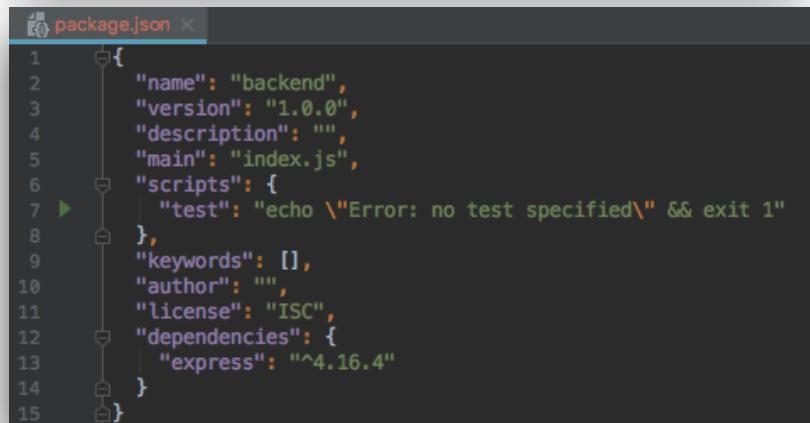
```
>> cd backend
```

```
>> npm init -y (creates package.json)
```

```
>> npm install express --save
```



```
package.json
1 {  
2   "name": "backend",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\"$Error: no test specified\\\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC"  
12}  
13
```

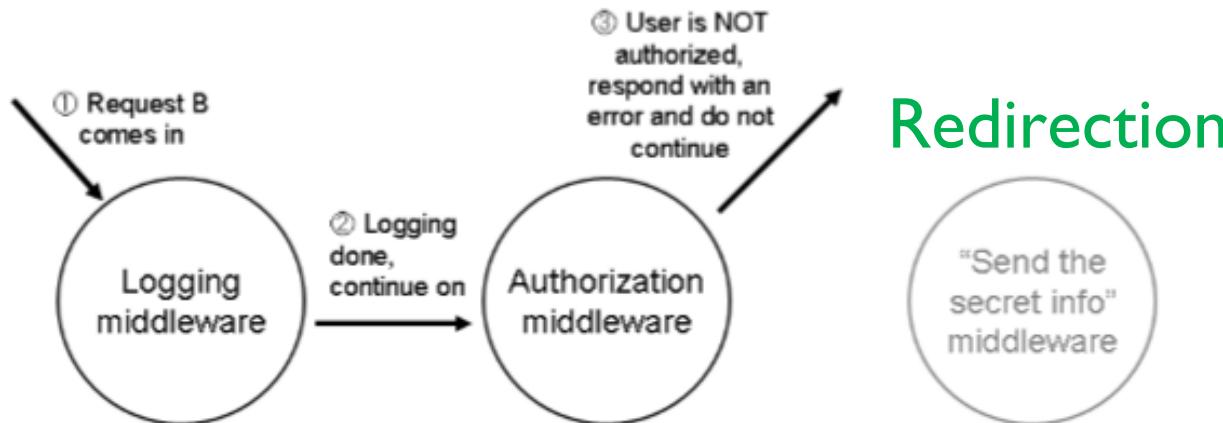
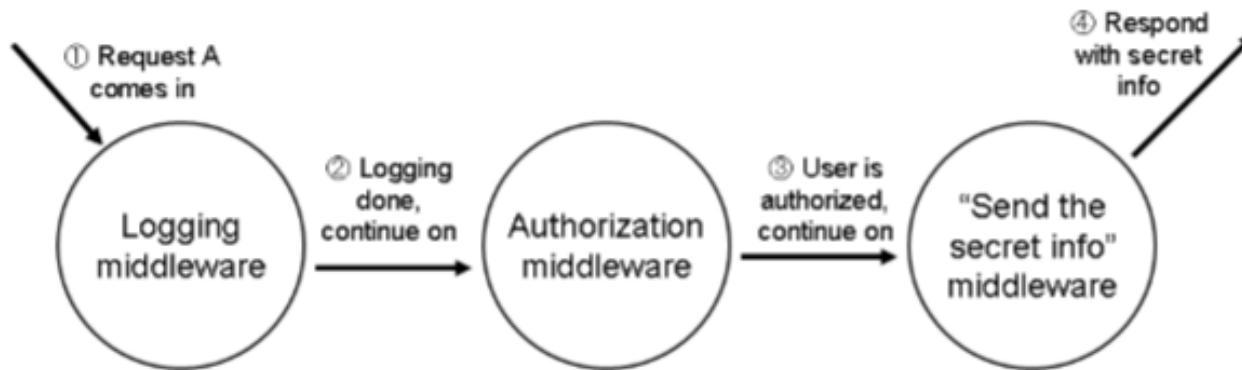


```
package.json
1 {  
2   "name": "backend",  
3   "version": "1.0.0",  
4   "description": "",  
5   "main": "index.js",  
6   "scripts": {  
7     "test": "echo \\"$Error: no test specified\\\" && exit 1"  
8   },  
9   "keywords": [],  
10  "author": "",  
11  "license": "ISC",  
12  "dependencies": {  
13    "express": "^4.16.4"  
14  }  
15}  
16
```



Middleware

```
app.put('/logout', isLoggedIn, logout)  
function isLoggedIn(req, res, next)
```



Routing with Express

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

var app = express()

app.get('/', getIndex)
app.post('/', postIndex)

function getIndex(req, res) {
    res.send('hello world!')
}

function postIndex(req, res) {
    res.send('You POSTed to the homepage')
}
```

```
const port = process.env.PORT || 3000
const server = app.listen(port, () => {
    const addr = server.address()
    console.log(`Server listening at http://${addr.address}:${addr.port}`)
})
```



Preprocess Request Body

- Must preprocess req body
 - chunk data
- Would be better to have someone do that for us
 - can use middleware

```
1  var http = require('http');
2
3  var host = '127.0.0.1';
4  var port = 3333;
5
6  http.createServer(preprocess).listen(port, host);
7  console.log('Server running at http://' + host + ':' + port);
8
9  function preprocess(req, res) {
10    var body = '';
11    req.on('data', function(chunk) {
12      body += chunk
13    });
14    req.on('end', function() {
15      req.body = body;
16      server(req, res)
17    })
18  }
```



Install Middleware (body-parser)

```
>> npm install body-parser --save
```

```
2 const express = require('express')
3
4
5 const addArticle = (req, res) => {
6     console.log('Payload received', req.body)
7     res.send(req.body)
8 }
9
10 const hello = (req, res) => res.send({ hello: 'world' })
11
12 const app = express()
13
14 app.post('/article', addArticle)
15 app.get('/', hello)
16
17 const port = process.env.PORT || 3000
18 const server = app.listen(port, () => {
19     const addr = server.address()
20     console.log(`Server listening at http://${addr.address}:${addr.port}`)
21 })
```

Server listening at http://:::3000
Payload received undefined

```
mjoyner@FVFG21ZAQ6LT ~ % curl -H 'Content-Type: application/json' -d '{"text":"This is my new article"}' http://127.0.0.1:3000/article
```



Accepting JSON Payloads

```
>> npm install body-parser --save
```

```
2 const express = require('express')
3 const bodyParser = require('body-parser')
4
5 const addArticle = (req, res) => {
6     console.log('Payload received', req.body)
7     res.send(req.body)
8 }
9
10 const hello = (req, res) => res.send({ hello: 'world' })
11
12 const app = express()
13 app.use(bodyParser.json())
14 app.post('/article', addArticle)
15 app.get('/', hello)
16
17 const port = process.env.PORT || 3000
18 const server = app.listen(port, () => {
19     const addr = server.address()
20     console.log(`Server listening at http://${addr.address}:${addr.port}`)
21 })
```

Server listening at http://:::3000
Payload received { text: 'This is my new article' }

```
mjoyner@FVFG21ZAQ6LT ~ % curl -H 'Content-Type: application/json' -d '{"text":"This is my new article"}' http://127.0.0.1:3000/article
```



Express JSON Payloads

- Node uses `require` to load modules
- Express `use` is applied before every endpoint `subsequently` defined
- No longer need `JSON.parse()` for `req.body`
- Passes request, response objects to function



Request Params

- Implement GET /articles/:id, not

- [/articles/1](#)
- [/articles/2](#)
- [/articles/3](#)

```
app.get('/article/:id', getArticle);
```

- The `:id` is a parameter whose value is determined at runtime
- Get the value from request
(e.g. `req.params.id`)



Create a New Post

Attempt to create a post with a unique id:

```
let post = { posts.length: req.body };
```

Error: Can't perform field access for object key

```
let post = {};  
post[posts.length] = req.body;
```

Use dictionary approach to access post by id using length

