DATA SERVER
(MONGODB)

DAY 2

Phonbopit (Chai) Sahakitchatchawan

— Writer @Devahoy

Chai Phonbopit

https://devahoy.com

Phonbopit

JS

express

node JS®

mongoDB

JS

Variables

```
// Declare variables.
var name = 'Chai';
let firstname = 'Phonbopit';

// Declare constant variable.
const country = 'Thailand';
```

Data Types

```
let foo = 42;     // Number
let foo = 'bar'; // String
let foo = true;   // Boolean
let foo = { name: 'Chai' } // Object
let foo = [1, 2, 3, 4] // Array
let foo;   // undefined
```

String

```
let name = 'Chai';
let blog = "devahoy";
let firstname = 'Phonbopit';

// Normal
console.log('Hello ' + firstname + '!');
// Hello Phonbopit!

// Template Literal
console.log(`Hello ${firsname}!`);
// Hello Phonbopit!
```

Operators

```
1 + 1 // 2
2 - 2 // 0
2 * 2 // 4
3 / 3 // 1
2 % 2 // 0

// logical
true || false // true
true && false // false

let number = '10';
number == 10   // true
number === 10 // false
```

If Condition

```
let score = 85;

if (score > 80) {
  console.log('Congrats! You got A');
}
```

If/Else Condition

```
let score = 85;

if (score > 80) {
  console.log('Congrats! You got A');
} else if (score < 50) {
  console.log('T_T You got F');
}
```

Switch Case

```
let score = 100;

switch(score) {
  case 100:
    console.log('Perfect!');
    break;
  case 50:
    console.log('Alive!');
    break;
  default:
    console.log('Try again!');
}
```

For Loop

```
let counter = 10;

for (let i = 0; i < counter; i++) {
  console.log(i)
}
```

Function

```
// Function declaration
function sayHi() {
  return 'Hello World';
}

sayHi(); // "Hello World"
```

Function

```
// Function expressions
let sayHi = function() {
  return 'Hello World';
}

sayHi(); // "Hello World"
```

Function

```
// Arrow function
let sayHi = () => {
  return 'Hello World';
}

sayHi(); // "Hello World”
```

Array

```
let numbers = [1, 2, 3, 4, 5];

numbers[0];  // 1
numbers[4];  // 5

// Add value
numbers.push(10); // [1, 2, 3, 4, 5, 10]

// Remove value
numbers.pop();  // [1, 2, 3, 4, 5]
```

Array (Loop)

```
let numbers = [1, 2, 3, 4, 5];

for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}
```

Array (forEach)

```
let numbers = [1, 2, 3, 4, 5];

numbers.forEach(function(num) {
  console.log(num);
});
```

Array (map)

```
let numbers = [1, 2, 3, 4, 5];

const result = numbers.map(function(num) {
  return num * 2;
});

console.log(result);
// [2, 4, 6, 8, 10]
```

Array (filter)

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8];

const result = numbers.filter(function(num) {
  return num % 2 === 0;
});

console.log(result);
// [2, 4, 6, 8]
```

Array (reduce)

```
let numbers = [1, 2, 3, 4, 5, 6, 7, 8];

const result = numbers.reduce(function(sum, value) {
  return sum + value;
});

console.log(result); // 36
```

Object

```
let player = {};

player.name = 'Lionel Messi';
player.age = 30;

console.log(player);
// { name: 'Lionel Messi', age: 30 }
```

Object

```
let player = {
  name: 'Lionel Messi',
  age: 30
};

console.log(player);
// { name: 'Lionel Messi', age: 30 }
```

Object

```
let player = {
  name: 'Lionel Messi',
  age: 30
};

console.log(player.name);
// Lionel Messi

console.log(player['name']);
// Lionel Messi
```

Object

```
let player = {
  name: 'Lionel Messi',
  age: 30,
  say: function() {
    console.log('I am groot!!!');
  }
};

console.log(player.say());
// I am groot!!!
```

Scope

```
var x = 1;

console.log('1:', x); // 1: ?

if (true) {
  var x = 2;
  console.log('2:', x); // 2: ?
}

console.log('3:', x); // 3: ?
```

Scope

```
// Global Scope
var name = 'Chai';

function someFunction() {
    // Local Scope
    var name = 'Messi';
    console.log(name);
}

// Global Scope
function awesomeFunction() {
    var name = 'Ronaldo';
    console.log(name);
}

console.log(name);   // Chai
someFunction();      // Messi
awesomeFunction();   // Ronaldo
```

Scope

```
if (true) {
  // global scope
  var firstname = 'Cristiano';
  // local scope
  let lastname = 'Ronaldo';
}

console.log(firstname);
// Cristiano

console.log(lastname);
// ReferenceError: lastname is not defined
```

Hoisting

```
console.log(name);  // undefined;

var name = 'Chai';
```

Hoisting

```
var name;

console.log(name);  // undefined;

name = 'Chai';
```

Hoisting

```
var name;

console.log(name);   // undefined;

name = 'Chai';
```

Hoisting

```
var name;

console.log(name);   // undefined;

name = 'Chai';
```

Hoisting

```
function sayHi() {
  console.log(message);
  var message='Hello World'; //undefined
}

sayHi();
```

Hoisting (Function)

```
sayHi();

function sayHi() {
  console.log('Hello World'); // Hello World
}
```

Hoisting (Function)

```
sayHi();

let sayHi = function() {
  console.log('Hello World');
  // ReferenceError: sayHi is not defined
}
```

node

JS®

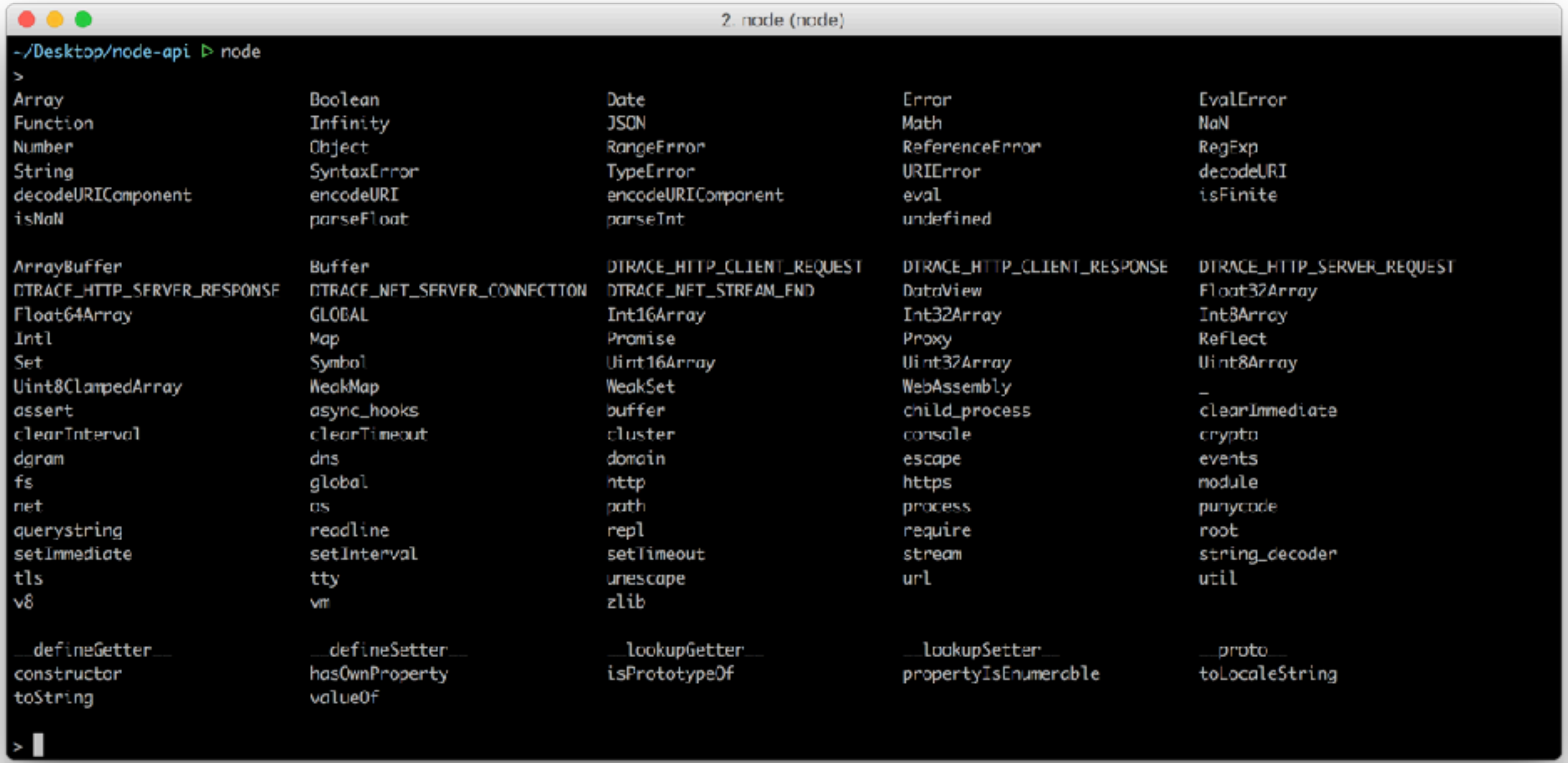
What’s Node.js

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

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

What’s Node.js

- JavaScript Runtime
- Run on Server
- Non-blocking I/O

Node REPL



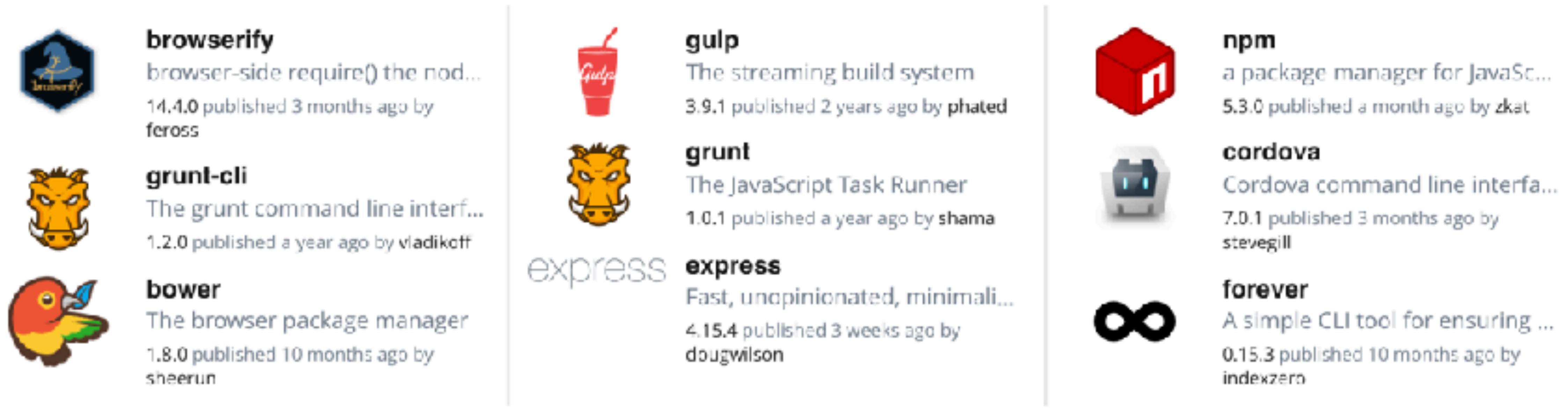
2. node (node)

```
~/Desktop/node-api ▷ node
>
Array                         Boolean                       Date                          Error                         EvalError
Function                      Infinity                      JSON                          Math                          NaN
Number                        Object                        RangeError                    ReferenceError                RegExp
String                        SyntaxError                   TypeError                     URIError                      decodeURI
decodeURIComponent            encodeURI                     encodeURIComponent            eval                          isFinite
isNaN                         parseFloat                    parseInt                      undefined

ArrayBuffer                   Buffer                        DTRACE_HTTP_CLIENT_REQUEST    DTRACE_HTTP_CLIENT_RESPONSE   DTRACE_HTTP_SERVER_REQUEST
DTRACE_HTTP_SERVER_RESPONSE   DTRACE_NET_SERVER_CONNECTION  DTRACE_NET_STREAM_END         DataView                      Float32Array
Float64Array                  GLOBAL                        Int16Array                    Int32Array                    Int8Array
Intl                          Map                           Promise                       Proxy                         Reflect
Set                           Symbol                        Uint16Array                   Uint32Array                   Uint8Array
Uint8ClampedArray             WeakMap                       WeakSet                       WebAssembly                   _
assert                        async_hooks                   buffer                        child_process                 clearImmediate
clearInterval                 clearTimeout                  cluster                       console                       crypto
dgram                         dns                           domain                        escape                        events
fs                            global                        http                          https                         module
net                           os                            path                          process                       punycode
querystring                   readline                      repl                          require                       root
setImmediate                  setInterval                   setTimeout                    stream                        string_decoder
tls                           tty                           unescape                      url                           util
v8                            vm                            zlib

__defineGetter__              __defineSetter__              __lookupGetter__              __lookupSetter__              __proto__
constructor                   hasOwnProperty                isPrototypeOf                 propertyIsEnumerable          toLocaleString
toString                      valueOf

> ▌
```

Node REPL

```
$ node
> console.log('Hello Node.js');
Hello Node.js
undefined
> function sayHi() { return "Hi!"; }
undefined
> sayHi();
'Hi!'
> 10 + 10
20
>
```

NPM



browserify
browser-side require() the nod...
14.4.0 published 3 months ago by feross

grunt-cli
The grunt command line interf...
1.2.0 published a year ago by vladikoff

bower
The browser package manager
1.8.0 published 10 months ago by sheerun

gulp
The streaming build system
3.9.1 published 2 years ago by phated

grunt
The JavaScript Task Runner
1.0.1 published a year ago by shama

express
Fast, unopinionated, minimali...
4.15.4 published 3 weeks ago by dougwilson

npm
a package manager for JavaSc...
5.3.0 published a month ago by zkat

cordova
Cordova command line interfa...
7.0.1 published 3 months ago by stevegill

forever
A simple CLI tool for ensuring ...
0.15.3 published 10 months ago by indexzero

Node Module

```
// http module
const http = require('http');

// path module
const path = require('path');

// file system module
const fs = require('fs');

// express module on npmjs.com
const express = require('express');
```

NPM Install

```
// Install local into node_modules
npm install express

// Install local and save to package.json
npm install express --save

// Install global
npm install express -g
```

package.json

```
// package.json
{
  "name": "awesome-module",
  "version": "1.0.0",
  "scripts": {
    "start": "node app.js",
    "dev": "nodemon app.js"
  }
}
```

```
// run start script
npm start
// run dev script
npm run dev
```

Export

```
// lib.js
function sum(a, b) {
  return a + b;
}

module.exports = {
  sum: sum
}
```

Import

```
// main.js
const sum = require('./lib').sum;

console.log(sum(5, 5)) // 10
```

Callback

```
setTimeout(function() {
  console.log('World');
}, 2000);

// Equal to
let callback = function() {
  console.log('World');
}

setTimeout(callback, 2000);
```

Callback

```
setTimeout(function() {
  console.log('World');
}, 2000);

console.log('Hello');

// Hello
// World
```

Callback

```
setTimeout(function() {
  console.log('World');
}, 2000);

console.log('Hello');

// Hello
// World
```

Callback

```
setTimeout(function() {
  console.log('World');
}, 2000);

console.log('Hello');

// Hello
// World
```

Callback

```
setTimeout(function() {
  console.log('World');
}, 2000);

console.log('Hello');

// Hello
// World
```

Callback (2)

```
let sayHi = function(word, callback) {
  if (word !== 'World') {
    return callback('Invalid Input');
  } else {
    return callback(null, `Hello ${word}`);
  }
}

sayHi('World!', function(err, data) {
  if (err) {
    console.log('err', err);
  } else {
    console.log('data', data);
  }
});
```

Callback (2)

```
let sayHi = function(word, callback) {
  if (word !== 'World') {
    return callback('Invalid Input');
  } else {
    return callback(null, `Hello ${word}`);
  }
}

sayHi('World!', function(err, data) {
  if (err) {
    console.log('err', err);
  } else {
    console.log('data', data);
  }
});
```

Callback (2)

```
let sayHi = function(word, callback) {
  if (word !== 'World') {
    return callback('Invalid Input');
  } else {
    return callback(null, `Hello ${word}`);
  }
}

sayHi('World!', function(err, data) {
  if (err) {
    console.log('err', err);
  } else {
    console.log('data', data);
  }
});
```

Callback (2)

```
let sayHi = function(word, callback) {
  if (word !== 'World') {
    return callback('Invalid Input');
  } else {
    return callback(null, `Hello ${word}`);
  }
}

sayHi('World!', function(err, data) {
  if (err) {
    console.log('err', err);
  } else {
    console.log('data', data);
  }
});
```

Promise

```
const promise = new Promise(function(resolve, reject) {
  if (code === 200) {
    resolve({ message: 'OK' });
  } else {
    reject({error: 'Something went wrong!'});
  }
});
```

Promise

```
callFunction()
.then()
.catch();
```

Promise (then)

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(200)
  .then(data => {
    console.log(data); // { message: 'OK' }
  })
  .catch(error => {
    console.log(error);
  });
```

Promise (then)

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(200)
  .then(data => {
    console.log(data); // { message: 'OK' }
  })
  .catch(error => {
    console.log(error);
  });
```

Promise (then)

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(200)
  .then(data => {
    console.log(data); // { message: 'OK' }
  })
  .catch(error => {
    console.log(error);
  });
```

Promise (catch)

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(500)
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.log(error); // { error: 'Something went wrong' }
  });
```

Promise (catch)

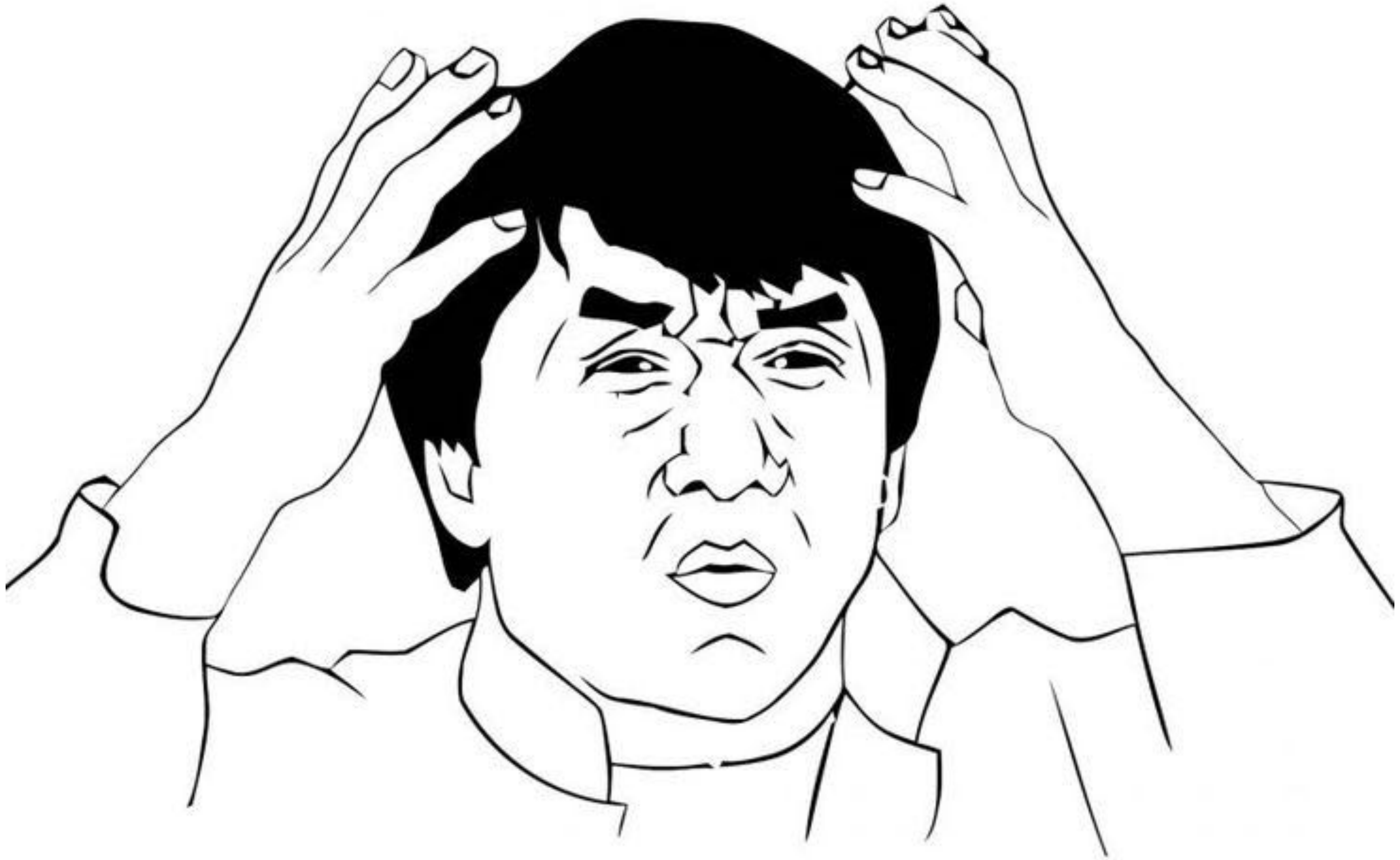
```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(500)
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.log(error); // { error: 'Something went wrong' }
  });
```

Promise (catch)

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

getStatus(500)
  .then(data => {
    console.log(data);
  })
  .catch(error => {
    console.log(error); // { error: 'Something went wrong' }
  });
```

Async/Await

```
function getStatus(code) {
  return new Promise((resolve, reject) => {
    if (code === 200) {
      resolve({ message: 'OK' });
    } else {
      reject({error: 'Something went wrong!'});
    }
  });
}

async function getAsyncStatus() {
  try {
    const data = await getStatus(200)
    console.log(data)
  } catch (error) {
    console.log(error)
  }
}

getAsyncStatus()
```



express

Hello World

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

https://expressjs.com

Hello World

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

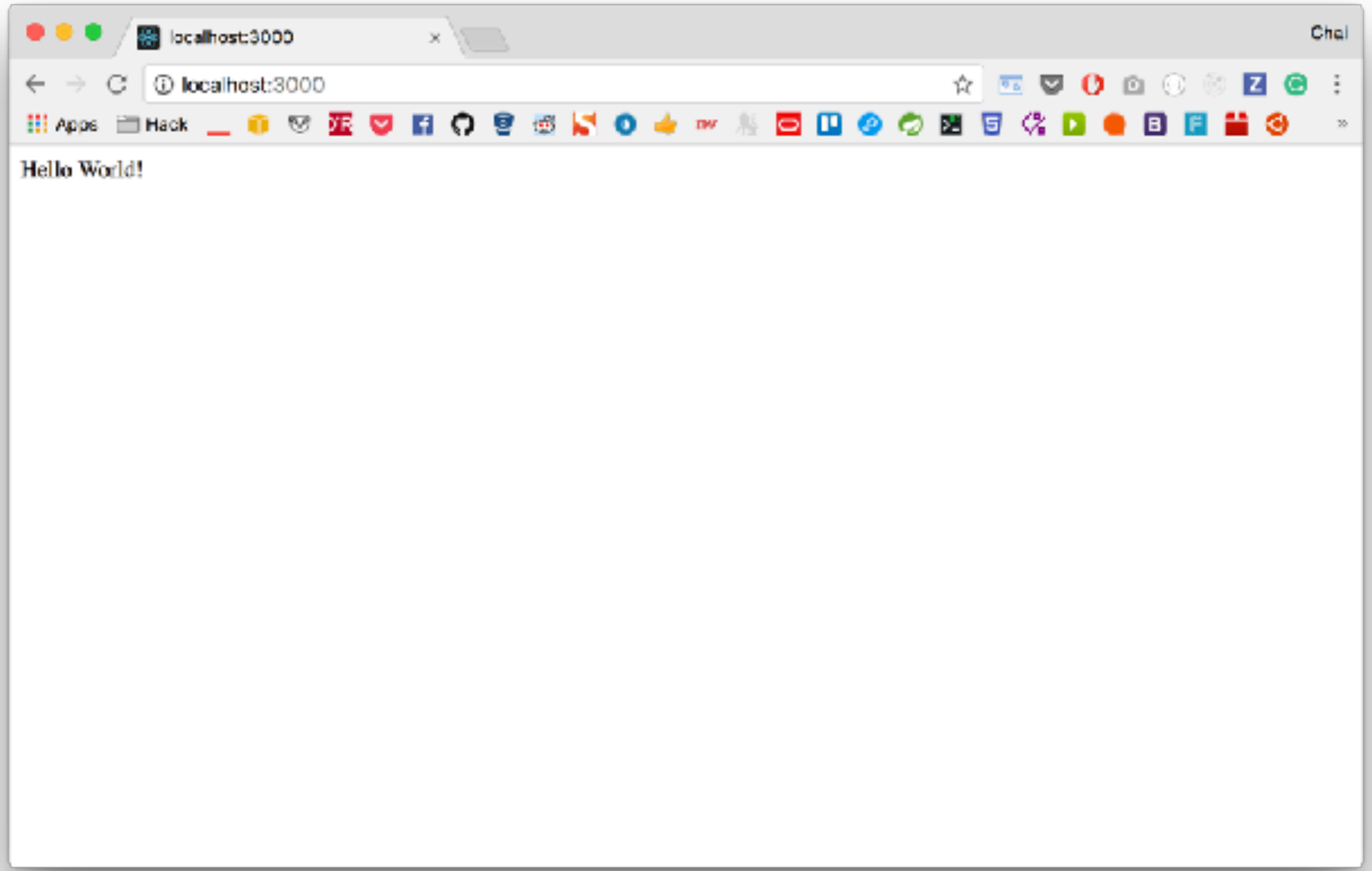
Hello World

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Hello World

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

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

Hello World



Hello World!

```
$ node server.js
Example app listening on port 3000!
```

Routing

```
app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.post('/', function (req, res) {
  res.send('Got a POST request');
});

app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user');
});

app.delete('/user', function (req, res) {
  res.send('Got a DELETE request at /user');
});
```

Request params

```
Route path: /users/:id/items/:itemId
Request URL: http://localhost:3000/users/1/items/900
req.params: { "id": "1", "itemId": "900" }

app.get('/users/:id/items/:itemId', function (req, res) {
  res.send(req.params);
});
```

Request query

```
GET /search?q=devahoy
req.query.q
// => "devahoy"

GET /users?order=desc&limit=30
req.query.order
// => "desc"
req.query.limit
// => "30"
```

Request body

```
const app = require('express')();
const bodyParser = require('body-parser');

// for parsing application/json
app.use(bodyParser.json());
// for parsing application/x-www-form-urlencoded
app.use(bodyParser.urlencoded({ extended: true }));

app.post('/users', function (req, res) {
  console.log(req.body);
  res.json(req.body);
});
```

Response

```
res.send('OK');
res.status(400).send('Bad Request');
res.json({ message: 'Hello World' });
```

Simple API

```
// package.json

{
  "name": "simple-api",
  "server": "1.0.0",
  "dependencies": {
    "body-parser": "^1.17.2",
    "express": "^4.15.4"
  }
}
```

Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get('/hello/:name', function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get('/hello/:name', function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get(‘/hello/:name’, function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get('/hello/:name', function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

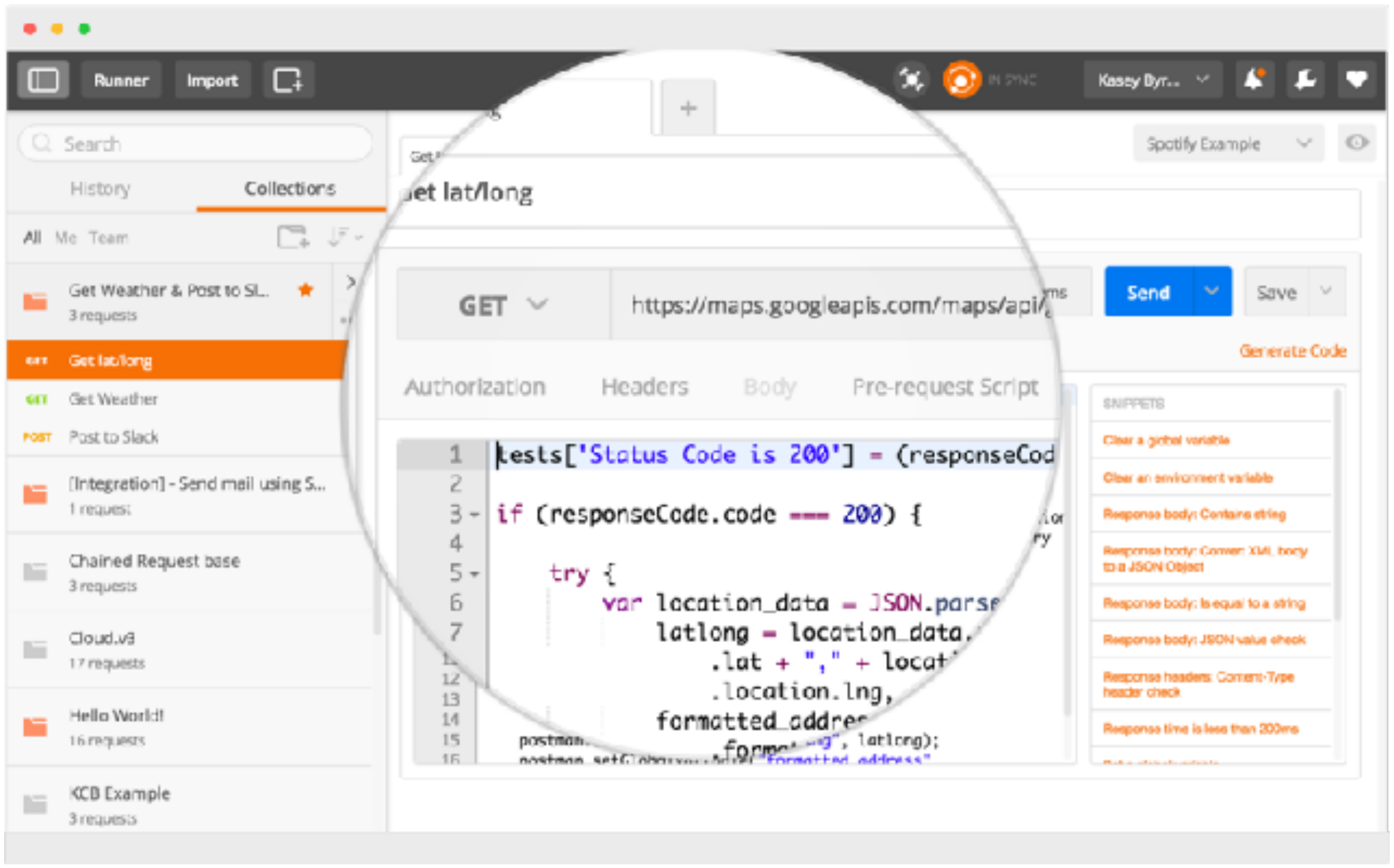
Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get('/hello/:name', function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

Simple API

```
// app.js

const express = require('express');
const bodyParser = require('body-parser');
const app = express();

app.use(bodyParser.urlencoded({ extended: true }));
app.use(bodyParser.json());

const router = express.Router();

router.get('/', function(req, res) {
  res.json({ message: 'It\s work!' });
});

router.get(‘/hello/:name’, function(req, res) {
  res.json({ data: req.params.name });
});

app.use('/api', router);

app.listen(8888, function() {
  console.log('Simple API listening on port 8888!');
});
```

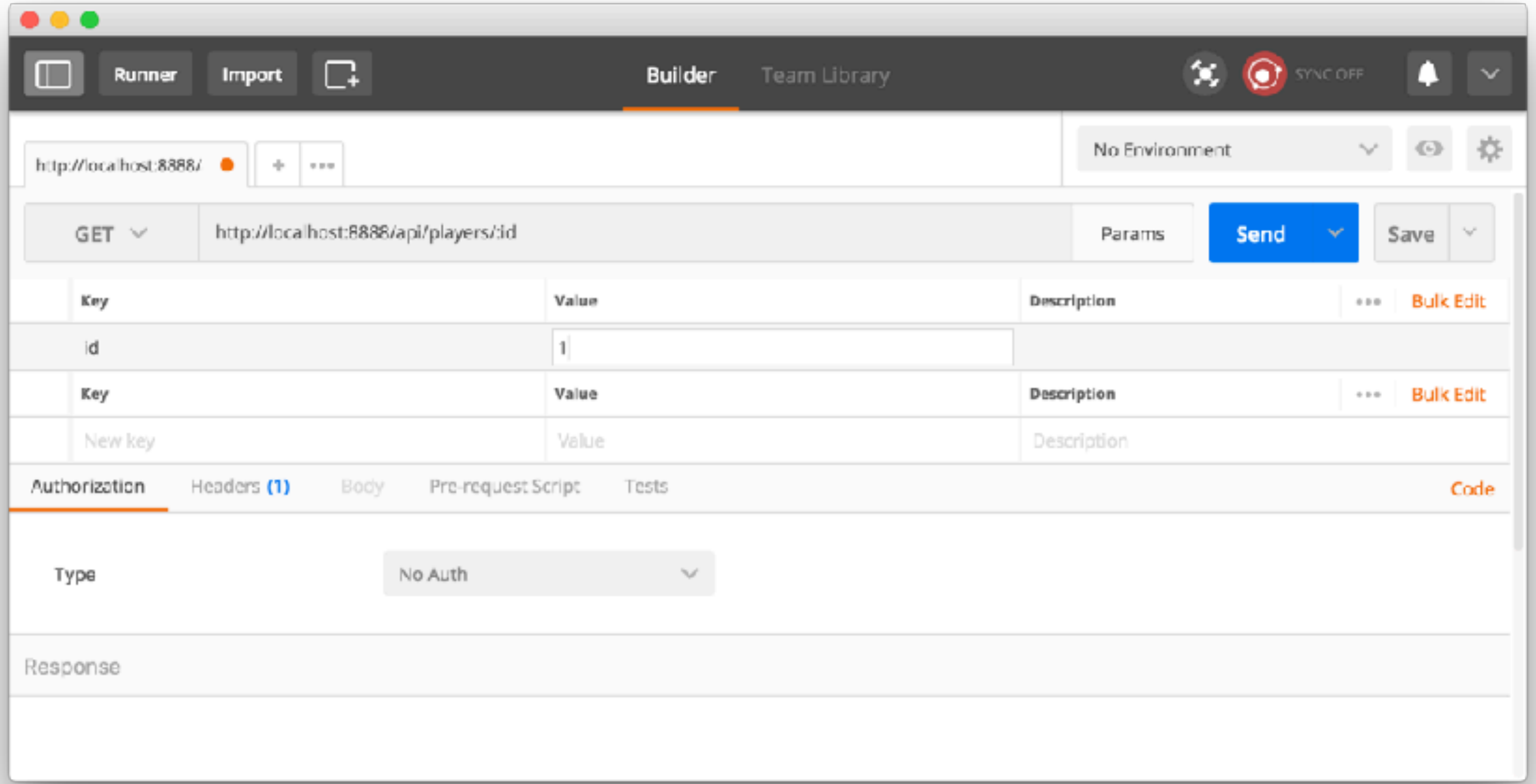
What's Postman?



```
tests['Status Code is 200'] = (responseCod

if (responseCode.code === 200) {

    try {
        var location_data = JSON.parse
            latlong = location_data.
                .lat + "," + locat
                .location.lng,
            formatted_addres
                format...g", latlong);
postman...
postman.setGlobalVar... "formatted_address"
```

https://www.getpostman.com/

GET



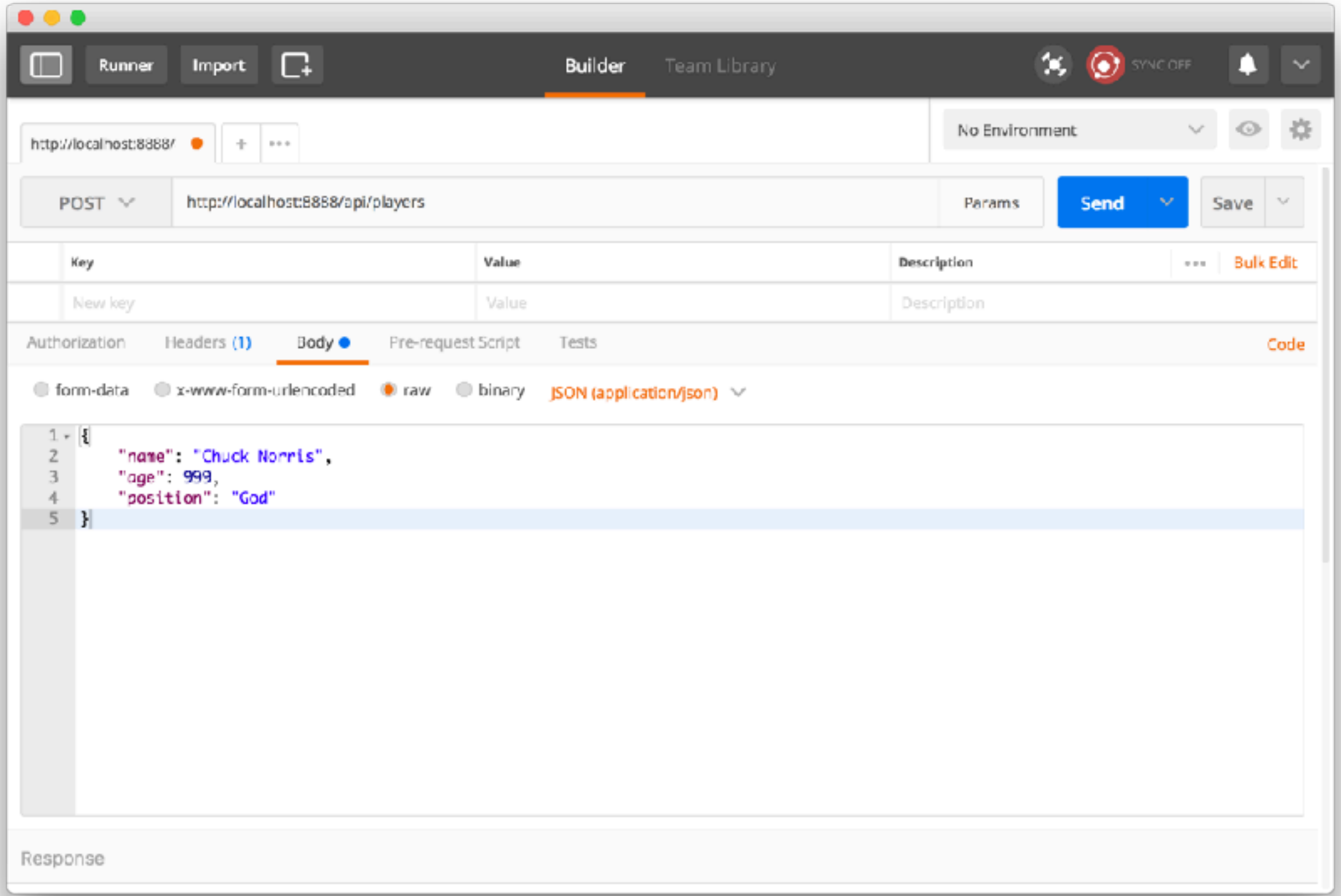
GET http://localhost:8888/api/players/:id

Key	Value	Description
id	1	

Key	Value	Description
New key	Value	Description

Type: No Auth

POST



POST http://localhost:8888/api/players

Key	Value	Description
New key	Value	Description

Body: form-data, x-www-form-urlencoded, [✓ raw], binary — JSON (application/json)

```
{
    "name": "Chuck Norris",
    "age": 999,
    "position": "God"
}
```

mongoDB

Mongoose

```
const mongoose = require('mongoose')
mongoose.connect('mongodb://localhost/test')

const Student = mongoose.model('Student', { name: String })

const bobby = new Student({ name: 'Bobby' })
bobby.save().then(() => console.log('ok'))
```

Schema

```
const mongoose = require('mongoose')
const Schema = mongoose.Schema

let Student = new Schema({
  name: String,
  score: Number
})

module.exports = mongoose.model('Student', Student)
```

Resource

```
GET     /students     : List all students
POST    /students     : Create a new student
GET     /students/1   : Show student with id 1
PUT     /students/1   : Update student with id 1
DELETE  /students/1   : Delete student with id 1
```

HTTP Status Codes

```
200 : OK
201 : Created
400 : Bad Request
401 : Unauthorized
403 : Forbidden
429 : Too Many Request
500 : Internal Server
```

Mongoose (Query)

```
const Student = require('../../models/student')

router.get('/students', (req, res) => {
  Student.find().then(data => {
    res.json(data)
  }).catch(error => {
    res.status(400).json(error)
  })
})
```

Mongoose (Query 2)

```
const Student = require('../../models/student')

router.get('/students/:id', (req, res) => {
  const id = req.params.id

  Student.findById(id) // equivalent to findOne({ _id: id })
  .then(data => {
    res.json(data)
  })
  .catch(error => {
    res.status(400).json(error)
  })
})
```

Mongoose (Create)

```
const Student = require('../../models/student')

router.post('/students', (req, res) => {
  let payload = req.body

  let student = new Student({
    name: payload.name,
    score: payload.score
  })

  student.save().then(data => {
    res.json(data)
  }).catch(error => {
    res.status(400).json(error)
  })
})
```

Mongoose (Update)

```
const Student = require('../../models/student')

router.put('/students/:id', (req, res) => {
  const id = req.params.id

  const payload = req.body

  Student.findByIdAndUpdate(id, {
    $set: {
      name: payload.name,
      score: payload.score
    }
  }).then(data => {
    res.json(data)
  }).catch(error => {
    res.status(400).json(error)
  })
})
```

Mongoose (Delete)

```
const Student = require('../../models/student')

router.delete('/students/:id', (req, res) => {
  const _id = req.params.id

  Student.remove({ _id }).then(data => {
    res.json(data)
  }).catch(error => {
    res.status(400).json(error)
  })
})
```

YOUR TURN!