# General Properties

## Execute code

|  |
| --- |
| node C:\git\node-and-mysql\test.js |

## Install Package

|  |
| --- |
| npm install package\_name |

Installs package in the current directory

## JavaScript interpreter

"node" Works as an interpreter

|  |
| --- |
| node |

Output:

|  |
| --- |
| > Math.random()  0.30066378795211857v12.16.1.  >  >  > 1  1  > 1+1  2  > a="qwerty"  'qwerty'  > a  'qwerty'  > a=a+a  'qwertyqwerty'  > Math.floor((Math.random() \* 10) + 1)  6 |

## JavaScript PRINTF Equivalent

Use ${varname} inside backquote string ``

|  |
| --- |
| > a={ f1: "alex" }  { f1: 'alex' }  > ` my name is ${a.f1} `  ' my name is alex ' |

## Faker

Package that generates tonnes of fake data. Install:

|  |
| --- |
| C:\git\node-and-mysql>npm install faker  npm WARN saveError ENOENT: no such file or directory, open 'C:\git\node-and-mysql\package.json'  npm notice created a lockfile as package-lock.json. You should commit this file.  npm WARN enoent ENOENT: no such file or directory, open 'C:\git\node-and-mysql\package.json'  npm WARN node-and-mysql No description  npm WARN node-and-mysql No repository field.  npm WARN node-and-mysql No README data  npm WARN node-and-mysql No license field.  + faker@5.1.0  added 1 package from 1 contributor and audited 1 package in 4.428s  found 0 vulnerabilities |

### Test Faker

|  |
| --- |
| // Check if faker is installed  var faker = require('faker');  //generate data  var data = [];  for(var i = 0; i < 500; i++){  data.push([  faker.internet.email(),  faker.date.past()  ]);  }  console.log(data); |

## MySQL

### Install Package

|  |
| --- |
| C:\git\node-and-mysql>npm install mysql  npm WARN saveError ENOENT: no such file or directory, open 'C:\git\node-and-mysql\package.json'  npm WARN enoent ENOENT: no such file or directory, open 'C:\git\node-and-mysql\package.json'  npm WARN node-and-mysql No description  npm WARN node-and-mysql No repository field.  npm WARN node-and-mysql No README data  npm WARN node-and-mysql No license field.  + mysql@2.18.1  added 11 packages from 15 contributors and audited 12 packages in 1s  found 0 vulnerabilities |

### Alter user to allow Node.js Login

|  |
| --- |
| ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql\_native\_password BY 'password';  flush privileges; |

### Test connection script

|  |
| --- |
| var mysql = require('mysql');  var connection = mysql.createConnection({  host : 'localhost',  user : 'root',  password : 'password',  database : 'join\_us'  });  connection.connect(function(err) {  if (err) throw err;  console.log("Connected!");  });  var q = 'select now() as sysdate';  connection.query(q, function(err, result, fields) {  if (err) throw err;  console.log(result);  });  connection.end(); |

Results:

|  |
| --- |
| C:\git\node-and-mysql>node mysql\_test.js  Connected!  [ RowDataPacket { sysdate: 2020-09-04T05:06:17.000Z } ] |

# Using BIND Variables

Note : It's actually Query Rewrite rather than use of bind variables. I.e. ? is replaced by values in SQL string.

## INSERT SET

The best way to insert one record us to user INSERT SET syntax with JavaScript record type

|  |
| --- |
| var person = {};  person.email = faker.internet.email();  var q = 'insert into users set ?';    connection.query(q, person, function(err, result, fields) {  console.log(`insert email ${person.email}`);  if (err) throw err;  console.log(result);  });  // Two fields  person.email = faker.internet.email();  person.created\_at = '2011-03-03';  connection.query(q, person, function(err, result, fields) {  console.log(`insert email ${person.email} and bd ${person.created\_at} `);  if (err) throw err;  console.log(result);  }); |

## INSERT "Values (?)" or INSERT "Values ?"

### Insert Values (?)

This is also used to insert one record ONLY

Bind variables must be in a form of **double array** [[ …],[ …], ]

|  |
| --- |
| Q = 'insert into users (email, created\_at ) values ( ? ) ';  person.email = faker.internet.email();  person.created\_at = faker.date.past();    sql\_stmt = connection.query(q, [[ person.email, person.created\_at]] , function(err, result, fields) {  console.log(`insert VALUES email ${person.email} and bd ${person.created\_at}`);  if (err) throw err;  console.log(result);  });  console.log ("Generated SQL statement 2 is:"+sql\_stmt.sql); |

### INSERT Values ?

Bind variables must be in a form of **triple**  **array** [[[ …],[ …], ]]

One record:

|  |
| --- |
| q = 'insert into users (email, created\_at ) values ? ';  person.email = faker.internet.email();  person.created\_at = faker.date.past();    sql\_stmt = connection.query(q, [[[ person.email, person.created\_at]]] , function(err, result, fields) {  console.log(`insert VALUES email ${person.email} and bd ${person.created\_at}`);  if (err) throw err;  console.log(result);  }); |

Many Records:

|  |
| --- |
| var data = [];  for(var i = 0; i < 500; i++){  data.push([  faker.internet.email(),  faker.date.past()  ]);  }  // console.log(data);  //Values ?  var q = 'INSERT INTO users (email, created\_at) VALUES ?';  var sql\_stmt = connection.query(q, [data], function(err, result) {  if (err) throw err;  console.log(result);  }); |

# Building Join Us App

Create new folder in mysql Repositorty

## Create folder

|  |
| --- |
| mkdir C:\git\mysql\JoinUs  pushd C:\git\mysql\JoinUs |

## NPM INIT

|  |
| --- |
| npm init |

Creates C:\git\mysql\JoinUs\package.json:

|  |
| --- |
| C:\git\mysql\JoinUs>npm init  This utility will walk you through creating a package.json file.  It only covers the most common items, and tries to guess sensible defaults.  See `npm help json` for definitive documentation on these fields  and exactly what they do.  Use `npm install <pkg>` afterwards to install a package and  save it as a dependency in the package.json file.  Press ^C at any time to quit.  package name: (joinus) join\_us  version: (1.0.0)  description: some cult  entry point: (index.js) app.js  test command:  git repository: akluev/mysql  keywords:  author:  license: (ISC)  About to write to C:\git\mysql\JoinUs\package.json:  {  "name": "join\_us",  "version": "1.0.0",  "description": "some cult",  "main": "app.js",  "scripts": {  "test": "echo \"Error: no test specified\" && exit 1"  },  "repository": {  "type": "git",  "url": "git+https://github.com/akluev/mysql.git"  },  "author": "",  "license": "ISC",  "bugs": {  "url": "https://github.com/akluev/mysql/issues"  },  "homepage": "https://github.com/akluev/mysql#readme"  }  Is this OK? (yes) y |

## Install and Save Packages

To save packages into package.json use –save

|  |
| --- |
| npm install faker mysql express --save |

Output:

|  |
| --- |
| C:\git\mysql\JoinUs>npm install faker mysql express --save  npm notice created a lockfile as package-lock.json. You should commit this file.  + faker@5.1.0  + mysql@2.18.1  + express@4.17.1  added 60 packages from 49 contributors and audited 60 packages in 5.726s  found 0 vulnerabilities |

## Create .gitignore

We need to exclude node\_modeules from git

On Windows:

|  |
| --- |
| echo node\_modules/\* >.gitignore |

# Hellow World in Express

<http://expressjs.com/>

## Create app.js

|  |
| --- |
| var express = require('express');  //load express  var app = express();    app.get("/", function(req, res){  res.send("HELLO FROM OUR WEB APP!");  });    app.listen(8080, function () {  console.log('App listening on port 8080!');  }); |

## Start app

|  |
| --- |
| node app |

Output

|  |
| --- |
| C:\git\mysql\JoinUs>node app  App listening on port 8080! |

To stop web service press Ctrl-C

## Test

Access app via <http://localhost:8080>

For example using curl:

|  |
| --- |
| curl -kis <http://localhost:8080>  HTTP/1.1 200 OK  X-Powered-By: Express  Content-Type: text/html; charset=utf-8  Content-Length: 23  ETag: W/"17-Qf2MHDbVM4ep8LY/7jT/xX8h618"  Date: Sat, 05 Sep 2020 03:07:17 GMT  Connection: keep-alive  HELLO FROM OUR WEB APP! |
| curl -kis <http://localhost:8080/1111>  HTTP/1.1 404 Not Found  X-Powered-By: Express  Content-Security-Policy: default-src 'none'  X-Content-Type-Options: nosniff  Content-Type: text/html; charset=utf-8  Content-Length: 143  Date: Sat, 05 Sep 2020 03:07:33 GMT  Connection: keep-alive  <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="utf-8">  <title>Error</title>  </head>  <body>  <pre>Cannot GET /1111</pre>  </body>  </html> |

# Add EJS

EJS allows to work with static HTML and add dynamic rendering

## Install EJS

C:\git\mysql\JoinUs>npm install ejs --save

+ ejs@3.1.5

added 15 packages from 8 contributors and audited 75 packages in 1.563s

found 0 vulnerabilities

## Create Views folder

mkdir views

## Create home.ejs

**Note: Use <%= … %> as parameters**

|  |
| --- |
| <h1>JOIN US</h1>    <p class="lead">Enter your email to join <strong><%= count %></strong>  others on our waitlist. We are 100% not a cult. </p>    <form method="POST" action='/register'>  <input type="text" class="form" name="email" placeholder="Enter Your Email">  <button>Join Now</button>  </form> |

## Register in app.js

|  |
| --- |
| //register ejs  app.set ("view engine", "ejs");  var connection = mysql.createConnection({  host : 'localhost',  user : 'root',  password : 'password',  database : 'join\_us'  });  … |

## Call home.ejs and pass a parameter

Pass parameter count

|  |
| --- |
| pp.get("/", function(req, res){  //console.log(req);  var q = 'select count(\*) cnt from users';  connection.query(q, function(err, result, fields) {  if (err) {  res.send("Error. Check logs");  throw err;  }  // console.log(result);  var cnt = result[0].cnt;  console.log(`User Count ${cnt}` );  res.render("home.ejs", { count: cnt, count2: cnt+1});  });  }); |

# Add Body Parser to Process POST Requests

Body parser allows to process POST requests that come in HTML body

## Install Body-Parser

C:\git\mysql\JoinUs>npm install body-parser --save

+ body-parser@1.19.0

updated 1 package and audited 76 packages in 1.023s

found 0 vulnerabilities

## Register with the app

Add registration in app.js

|  |
| --- |
| var express = require('express');  var mysql = require('mysql');  var bodyParser = require('body-parser');  //load express  var app = express();  //register ejs  app.set ("view engine", "ejs");  // register body parser  app.use (bodyParser.urlencoded({extended: true}) ); |

## Accessing Request Body

|  |
| --- |
| // Register  app.post("/register", function(req, res){  console.log("Request", req.body);  var num = Math.floor((Math.random() \* 10) + 1);  res.send("Your lucky number is " + num);  }); |

# Processing POST

## App.js handler

Note error handling

|  |
| --- |
| app.post("/register", function(req, res){  //console.log("Request", req.body);  var q="insert into users set ?";  var newuser = {email: req.body.email};  connection.query(q, newuser, function(err, result, fields) {  if (err) {  console.log(`Registration Error for ${newuser.email}\n `, err);  res.render("register\_error" , {email : req.body.email, error: err});  return 0;  }  res.render("register" , {email : req.body.email});  console.log(`Registration Success for ${newuser.email}`);  });  }); |

## Register.ejs

|  |
| --- |
| Thank you for registing your email <a href="mailto:<%= email %>"><%= email %></a> with us! <p>  We will be in touch shortly |

## Register\_error.ejs

Shows error message

|  |
| --- |
| Sorry, we cannot register email <a href="mailto:<%= email %>"><%= email %></a> with us! <p>  Reason <b><%= error.sqlMessage %> </b> |

# Styling

## Create Static Folder

mkdir statis

Put static CSS files ( app.css) in the static folder

## Register in app.js

Register static folder as following

|  |
| --- |
| var express = require('express');  var mysql = require('mysql');  var bodyParser = require('body-parser');  //load express  var app = express();  //register ejs  app.set ("view engine", "ejs");  // register body parser  app.use (bodyParser.urlencoded({extended: true}) );  // register static folder  app.use(express.static(\_\_dirname + "/static")); |

## Add references to CCS Views i.e. HTML files in view folder

|  |
| --- |
| <head>  <link href="https://fonts.googleapis.com/css?family=Roboto:100,300,400" rel="stylesheet">  <link rel="stylesheet" href="/app.css">  </head>  <p>Thank you for registing your email <a href="mailto:<%= email %>"><%= email %></a> with us!  <p>We will be in touch shortly |