**Server-Side Scripting**

Node.js

* Node.js is a JavaScript API that helps in server-side scripting.
* Node.js is the same as Java Servlet Pages or JSPs where you can also embed code inside and at the same time create html document.

Node.js Library

* Node.js’ package ecosystem is the node package manager (npm), which contains a wide array of open source libraries around the globe.

Installing Node.js and Using npm

* After installing node.js on the computer, its already fairly easy to add modules to your local project or global modules for all your project that uses node.js as a platform for server-side scripting.
* A package.json is typically used to specify modules and its dependencies when installing modules for your local node.js project.

A typical package.json usually looks something like this:

{

“name”: “node”,

“version”: “1.0.0”,

“scripts”: {

“start”: “node ./bin/www”

},

“dependencies”: {

“express”: “~4.12.4”

“mysql”: “0.1.0”

}

}

* Commands for npm

|  |  |
| --- | --- |
| Command | Description |
| npm init | Creates a package.json file with the initial its initial values/configurations |
| npm install | Installs your package.json file to your local or global project and all the dependencies specified in it. |
| npm install [module]  eg. npm install express | Installs the module that was specified. In this case install the module express in my project |
| npm install [module] --save  eg. npm install mysql --save | Installs the module that was specified. In this case install the module mysql and save that as a dependency inside a package.json file |

* Using modules in node.js

var variableName = require(“module”);

This will give the functions in the specified module to a variable and use those functions by using the name of the variable.

Eg.

var app = require(“express”);

var listener = app.listen(8888, function(){

console.log('Listening on port ' + listener.address().port);

});

This code will be using the express module and it will be listening for request coming from port 8888 and serves content and requests from there.

* Callback function

- Callback function are used for asynchronous processing and is declared as follows in node:

app.listen(‘data’, function([parameter(s)]){

//body

});

* Connecting to the Database

var mysql = require(“mysql”);

var connection = mysql.createConnection({

host: ‘localhost’,

user: ‘[username]’,

password: ‘[password]’,

database: ‘[databaseName]’

});

connection.connect();

* Querying from the Database

connection.query(“SELECT \* from [tableName]”, function(err, rows, fields) {

if (!err)

console.log(“The query is: “ rows);

else

console.log('Error while performing Query.');

});

The SQL query statement can also be replaced by DDL statements.

* Disconnecting from the Database

connection.end();

NOTE: Always close connections after opening one.

* Processing GET requests

var app = require(“express”);

app.get(‘/login’, function(request, response){

res.render(“login”);

});

The server will render the login page for the web application whenever it is requested in the url.

Eg. http://webapp/login

The login page will be given as a response

Without this code. The requested page will never render and an error page will be given as a response back to the requester.

* Creating html forms with node.js actions and HTTP Methods to generate a request.

\*On the HTML file

<form name=”form” action=”createUser” method=”post”>

<label for=”name”>Name: </label>

<input type=”text” name=”user” placeholder=”User Name”></input>

<input type=”submit” value=”Save”>

</form>

\*On the node.js file

var app = require(“http”);

var mysql = require(“mysql”);

var connection = mysql.createConnection({

host: ‘localhost’,

user: ‘[username]’,

password: ‘[password]’,

database: ‘[databaseName]’

});

connection.connect();

app.post('/createUser', function(req, res) {

var values = {user: req.body.user};

connection.query(“INSERT INTO [tableName] SET ?”, values, function(err, rows) {

if (!err)

res.redirect(“/success.js”);

else

console.log('Error while performing Query.');

});

});

connection.end();

This is the code for getting user input from the input box named user. After clicking the button save. It will then do the following codes specified in the app.post(“/createUser”, …) callback function. What the callback function will do is to save the value specified in the input box of the form to an array with the specific fields in the database (eg. var values = {user: req.body.user}) and then use those values to save it into the connected database and then redirect the user to a new page.

* Creating Sessions

var session = require('express-session');

var app = express();

app.use(session({

secret: 'lmao',

resave: false,

saveUninitialized: true,

}));

This creates a unique session for the user-agents.

* Embedding node.js codes in html files

<% This is way of specifying that within this tag are node.js codes %>

Eg.

\*On node.js file

app.get('/customers', function(req, res){

if(req.session.username){

var customers = [];

connection.query("SELECT \* FROM customer JOIN user ON custId = idNum WHERE status = '1' OR status='2'", function(err, rows){

if (err){ console.log("Error in query"); return; }

rows.forEach(function(item){

customers.push({

custId: item.custId,

firstName: item.firstName,

userName: item.userName,

lastName: item.lastName,

address: item.address,

email: item.email,

contactNumber: item.contactNumber,

status: item.status

});

});

res.render('customers', {customers: customers});

});

}else{

res.redirect('/login');

}

});

\*on the HTML file

<table class="table table-hover">

<tr>

<th>First name</th>

<th>Last name</th>

<th>User name

<th>Email</th>

<th>Contact number</th>

<th>Address</th>

<td>Status</td>

</tr>

<% customers.forEach(function(item){ %>

<tr>

<td><%= item.firstName %></td>

<td><%= item.lastName %></td>

<td><%= item.userName %></td>

<td><%= item.email %></td>

<td><%= item.contactNumber %></td>

<td><%= item.address %></td>

<td><%= item.status %></td>

<td>

</table>

What this does is to actually send the query as a customer array to a specific html file (customer.js) and then use that customer array as a way of getting the rows of the query and display it into the page using node.js code specified within the <% %> tag.