**FOUR PARTS OF API**

**CALLER: public/main.js**

var jar= {

hello: "hello"

}

$.post("/api/data", jar, function(data) {

console.log("return ", data)

})

$.get("/api/data/one/"+x, function (data) {

console.log(data);

**SERVER: app.js**

const express = require(‘express’)

const app = express()

const routes = require("./routes/api");

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

app.use(‘/api’, require(‘./routes/api’));

app.listen(…

**ROUTER: routes/api**

const express = require('express');

const router = = express.Router();

// any outside files that you want to route to..

const backlog = require("../backlog") // extra files

router.get("/data/one/:id", function(req, res) {

var id = req.params.id;

use id to make a call to database…

}

router.get(‘/ninjas’, function(req, res) {

var x = backlog.back(id)

// this passes data to function back in backlog file.

}

router.post(‘/ninjas’, function(req, res) {

var x = req.body.variable

// involve x in a calculation

Res.send(“hello”)

}

module.exports = router;

**EXTERIOR FILE** // uses exports

back = {

back: function(x){

console.log("back.js ", x)

return x;

}

}

module.exports = back;

**What is a module?**

A module encapsulates related code into a single unit of code .

Exports is an object, to which we can assign any number of properties.

We might have a file (greetings.js) that has a series of related functions (saved as “function expressions”).

// greetings.js

eng = function() {

return "Hello";

};

span = function() {

return "Hola";

};

You can add each function to module.exports individually:

exports.eng …

exports.span …

Now, module.exports reference the same object, but can carry several functions:

Module.exports = {

eng …

span …

}

Import the above

In new file, use require keyword. Doing so will put the two functions in the “greetings” keyword.

var greetings = require("./greetings.js");

// now we can access

greetings.eng()

**…. Module vs. module.exports**

Exports is an alias for module.exports.

Examples

exports.eng = function(y) {

console.log("you have this: ", y)

return 5

}

Same as

module.exports.eng = function(y) {

console.log("you have this: ", y)

return 5

}

However, exports itself, unlike module.exports, is never exported.

Here, we use a

This adds eng() to module.exports.

Module.exports = {

Eng(y) {

Console.log(“you have “, y)

Return 6

}

This wouldn’t work …

exports = {

Eng(y) {

Console.log(“you have “, y)

Return 6

}

TypeError: x.eng is not a function