**Call to API from back End**

The first step is to install the required Node.js package in project. Using the Node Package Manager (NPM), execute the following:

npm install request --save

Let’s say that we want issue a GET request and determine the IP address of the request, in this case our local machine. Within your project include the following:

var Request = require("request");

Request.get("http://httpbin.org/ip", (error, response, body) => {

if(error) {

return console.dir(error);

}

console.dir(JSON.parse(body));

});

The above snippet will issue a GET request and print out an error if it exists. Otherwise, if no error exists in the response, the response body will be parsed into a JavaScript object and then printed in the console.

Issuing GET requests is great, but what if you needed your application to write or change the remote data using a POST request or similar?

Let’s change our code to look like the following:

var Request = require("request");

Request.post({

"headers": { "content-type": "application/json" },

"url": "http://httpbin.org/post",

"body": JSON.stringify({

"firstname": "Nic",

"lastname": "Raboy"

})

}, (error, response, body) => {

if(error) {

return console.dir(error);

}

console.dir(JSON.parse(body));

});

Notice in the above snippet that we’ve changed our code to .post instead of .get. Most, not all, APIs expect requests to come in with a JSON body. For this reason we need to specify the body type as header information. The body cannot be sent as a JavaScript object so it must first be serialized into a string. Because we are explaining the format via a header, the receiving server will be able to parse the string back into an objec

**Write JSON to File**

JavaScript comes with the **JSON** class that lets you serialize an object to JSON with **JSON.stringify**. The file system **fs** module then writes data to the disk.

In the following, you’ll see examples for writing JSON to a file using callbacks and promises. Be careful with synchronous file operations in Node.js. The synchronous methods block the Node.js event loop and everything else has to wait for the file read/write.

const Fs = require('fs')

function writeToFile (data, path) {

const json = JSON.stringify(data, null, 2)

Fs.writeFile(path, json, (err) => {

if (err) {

console.error(err)

throw err

}

console.log('Saved data to file.')

})

}