**Ch. 20 – APIs:**

APIs (application programming interface) are a set of commands, functions, protocols and objects that programmers can use to create software (such as with JQuery) or interact with an external system (such as tinder accessing Facebook user’s likes/music choices).

API components:

Endpoint – the starting url to reach an API

Path – One of the branches of the root (endpoint). These are planned for ahead of time when designing a website.

Parameters – Placed at the end of the url. Custom messages sent by user, not planned for ahead of time.

<https://urlName.net/pathName?key1Name=customQuery1&key2Name=customQuery2>

Parameters separated by path with ?. Each key-value-pair is separated with &

Order of parameters does not matter.

Authentication – Every time an API is accessed, the developer and the frequency of requests is tracked. If the same developer meets a certain threshold of requests, they will start being billed for using this API.

Searching for specific parameters of API’s can get messy so we can use the program Postman to declutter which parameters and values are being used when accessing an API.

Information from APIs can be returned in JSON format (JS object notation), XML (extensible markup language) or HTML (hypertext markup language).

JSON format is similar to JS object’s in notation, where each object can hold keys with values, separated by commas. JSON name’s are wrapped in quotes, JS objects are not.

You can use the chrome extension “JSON viewer awesome” to transform raw API data into readable JSON style.

**How to make an external HTTP request:**

https is a native module of npm.

const https = require("https"); //no installation needed since it is native\*\*\*

https.get(URLInQuotes, function(response) {

console.log(response);

});

This sends a request to URLInQuotes and console.log’s the response.

response.on(“data”, function(data){

JSON.parse(data);

//code goes here

});

This waits for data to be sent, then runs the following anonymous function.

The hexadecimal from your HTTPS request is parsed into JSON format.

Add your res.send(codeHere) in the code section to display a response after your HTTPS request and data is retrieved.

JSON.parse(stringHere);

Turns string stringHere to JSON format.

Websites normally return data in string format.

JSON.stringify(JSobject);

Turns JS object JSobject to compact string.

**How to post (send) data to external website:**

https.request(url, options, function(response){ code here; }

this makes a request to the specified web sever with the options specified with the callback function specified.

Options can include the method type (get, post), authentication type (appID), etc.

In order to send that data, you must save your request in a constant and then use that constant request to send the JSON data (in string format\*\*\*). Must finish with request.end();

const request = https.request(url, options, function(response){ code here; }

request.write(JSONdata);

request.end();

You can use this to add users to an email listing on mailchimp

By checking the statusCode of your request, you can check if it was successful or not (200 is successful, anything else may not be)

Mailchimp wants your data to be formatted as the following:

const data = {

members: [ {

email\_address: email,

status: "subscribed",

merge\_fields: {

FNAME: firstName,

LNAME: lastName

} } ] }

**How to redirect someone:**

If you have a button on your failure page which posts to a different page on your server (“/failure”), you can have a post request which redirects the user back to your root page (“/”). This works effectively as a reset button and will return the user back to the sign-up screen.

**How to host your website using Heroku:**

app.listen(process.env.PORT || 3000, function() { code here } );

process.env.PORT – specifies a dynamic port that Heroku will choose later or the port 3000 when running locally.

You must include a ‘Procfile’ file in your project directory which tells Heroku how to start your web server. ‘Procfile’ must not have an extension. The contents must only include:

web: node appName.js

In your CLI, you must do the following to add your file to git.

git init

initializes the git repository

git add .

adds all your files to the git repository

git commit -m “Message goes here”

commits the current version with the message.

heroku create

Creates the Heroku file.

git push heroku master

Pushes the current version to the Heroku servers.

If you want to update another version, you must do your git add ., git commit, git push heroku master again.