FSW-115 Learn-It Notes

Week1

* The four basic CRUD operations are Create, Read, Update, & Delete.
* HyperText Transfer Protocol (HTTP) is an application-layer protocol that is used for fetching various resources.
* In programming we commonly use interfaces to abstract operations and provide a concise way to interact between a host and a user. An API stands for "Application Programming Interface" and is an interface that is usually expressed through shared protocols such as HTTP and relies on a standardized output format such as JSON.
* When your HTTP request is successful, you receive a status code of 200 OK. However, the meaning of success and response depends on the HTTP method used.
* JSON (JavaScript Object Notation) is commonly used to transfer data from a server to a client. The structure of JSON is best described as pairs of keys and values.
* If you are developing on a local server that means the server is most likely setup on your current machine.
* JSON.stringify() (data) is a method that converts a value or object to a string and additionally has 2 optional parameters that include replacer and space.
* When you type in youtube.com into a browser you are making a request to a server.
* IP addresses are the unique identifier for your computer that enables various websites to know where to send back data upon requesting it.
* To read, update, or delete a cookie you use the document.cookie property. Cookies are then commonly saved as name - value pairs.
* Cookies are a small file about a user that allows information to be carried over from related website or different sessions. These are commonly used because the server forgets everything once a connection is shut down.
* JSON.parse(string) primarily takes this data type as a parameter to construct a new JavaScript value or object.
* The client (such as my browser) interacts directly with a server when I make a web request. The server then interacts with a/an API to return data back to the client.
* If I wanted to get “Batman” as an output from the below code I would type

superHero.character

let superHero = {

“name” : "Bruce Wayne",

“character” : "Batman"

}

* If I run the following code car.year . I will get an error! This is because there is a trailing comma and therefore you are unable to parse all the values.

let car = {

“make” : "Toyota",

“model” : "Corolla",

“year” : "2011",

}

Week2

* Visit a web address such as https://www.youtube.com/feed/history. The history and feed in this example represent endpoints
* You know the following URL https://www.amazon.com/s?k=web+dev+books&ref=nb\_sb\_noss is an example of using a/an query string because there is a question mark followed by equals signs.
* Postman is a very popular API testing tool that allows you to send various types of requests and inspect their responses.
* The request header is a type of HTTP header that contains metadata such as the type of operating system you are using for the request. These types of headers can also contain response information such as whether access is allowed.
* A PUT HTTP request is used when you want to send data to the server.
* When I get a 500 internal server error that means that processing has failed on the server side due to unanticipated incident.
* When looking at a new API you should grab the base url plus one or two endpoints to understand the structure of the data in the response.
* In Postman you can get a Response Header message with the command responseHeaders.Message.
* One of the major differences between the PUT and the POST methods is that the results will always be the same for each operation.
* One of the major differences between the POST and the PUT methods is that the results will always be different for each operation.

Week3

* AJAX is not a programming language. However, it is very important to web development because it can update a web page without reloading the page.
* Every XMLHttpRequest object has a property called readyState that returns how far along in the request process the XMLHttpRequest client is. This can return one of five values numbering from 0 - 4. When the open method has been called readyState will return a value of 1.
* To create an XMLHttpRequest object that can be used to exchange data with a web server behind the scenes you set const xhr = new XMLHttpRequest().
* Synchronous XMLHttpRequest requests can oftentimes hang on the web because of their blocking nature. That is why asynchronous requests are preferred. In an asynchronous XMLHttpRequest you receive a callback when the data is received. This allows the browser to continue working while your request is being handled.
* The XMLHttpRequest.onreadystatechange property is set to a callback function because everytime the readyState updates the callback is triggered.
* The two properties that tell us our XMLHttpRequest has been successful and that our requested data is ready to use is the readyState and status. When successful and we are ready to handle our data these two properties will be return the values of 200 and 4.
* After you open your XMLHttpRequest, typically with the command xhr.open(params) where xhr represents the variable you set your XMLHttpRequest to, you then need to send the request which passes it along to the server.
* After making a XMLHttpRequest that is successful you will receive a response. To return the text of the response we call the responseText method on our XMLHttpRequest object. Now we are all ready to utilize our response.
* When we receive our response back as text from our XMLHttpRequest object we need to parse the response with JSON.parse() because our response text is a/an string.
* When you want to send data back to the database you will use a POST request. In order to provide this data along with our request we use the request body.
* If there is an item that I no longer want in my database I use the DELETE request to remove that item. Typically this request is accompanied by an id of that item.
* To retrieve a specific item or items from an api using Postman we set our request method to a GET request.
* If we want to update something in our database, say change the title of a book because it is misspelled, we would use a PUT request.
* In order to interact with a database we must first know the database model so we know the different properties and their data types for when we make our requests.
* The XMLHttpRequest method open() requires you pass it at least the following 2 parameters of method and url.

Week4

* Promises are similar to callbacks in that they are both an object that wraps an asynchronous operation and notifies you when it’s done.
* Promises are eager, meaning that it will start on a given task once the promise constructor is invoked.
* When sending a request to an API one of two things can happen. Either everything you requested goes well and the data you asked for is sent back or a/an error occurs and it is not.
* Promises have 3 states they can be in. The first state is pending which all promises start out in. Then a promise can either become resolved or rejected.
* The catch() method returns a Promise in the case of a request being rejected. This method can be chained in the same way as then().
* JavaScript promises do not expose the actual promise states. Only the function that is responsible for creating the promise will be aware of it’s status and have the ability to settle it with one of it’s two methods.
* Fetch is similar to a XMLHttpRequest object but expands on the capabilities with added features and the method is promise based.
* Using response.json() will parse an object into JavaScript and read the response as JSON.
* The response of a fetch() request is a stream, which is why when working with it we read the response as JSON since a promise is returned.
* Fetch’s default request method, which is called if only a url is passed as a parameter, is a get request.
* If I want to use Axios as a CDN I need to include it in a script tag either in the body or the head of my html.
* Axios is a promise-based http client that works with both your browser and node.js.
* One advantage that Axios has over other ways to make requests like fetch is that it automatically parse JSON into JavaScript.
* Using the method axios.get() can either return an array of items or one specific item depending on the way the api itself is setup.
* When working with an API and trying to retrieve just a single item with a request. It is common practice for an endpoint to be followed by a specific id of the item you are requesting like house/chores/……

Week5

* All Axios request methods require that a url is passed as the first parameter so that Axios knows where to send the request.
* The primary difference between make a post or put request versus say a get request in Axios is that you can include data.
* Each API will only save specific data. Databases really thrive on consistency. When we are sending data that means we need to make sure and follow the schema so that it is saved in a consistent manner.
* One of the perks of using Axios to send data is that we don’t need to convert a JavaScript object into JSON because it parses it for us.
* Since Axios is promise based all requests including post, put, and delete are resolved to either a then() or a/an catch().
* When working with Axios we are able to access the information and values inside our promise using response.data .
* If you are using a form to allow users to send data to your database with say an axios.post() request. It is common to tie the post request to the submit event on the form.
* If I would like to remove an item from a database I would use the axios.delete() request and typically pass in the specific id of the item on the endpoint.
* If an item is not in our database or we have removed it prior and we perform a request.get() on that id we will get back a/an empty string from res.data.
* When I want to update an item in a database using Axios I would use the axios.put() request and typically pass in the new data I would like to update it to.
* When making a request to update a database I will commonly need to provide the id along with a JavaScript object that includes the new values I wish to update.
* When doing a put or post request with Axios the second parameter we pass refers to the request body and generally looks something like axios.post(“someURL/….” , content).
* The Object.assign() copies the values from one or more source objects into a target object. It is commonly used when updating objects.
* If I want to add all the items I get back from a request to the DOM, I can loop through the list of items then use document.getElementById(‘’listID”).appendChild(p) to add them to an element such as a paragraph or h1.
* When making a post request and adding a list directly to the DOM, through like a click event, it is important to clear out the previous results or the complimenting get request will duplicate as you add new items.

Week6

* Promises have the ability to be returned to another promise and perform an additional operation on a resource which we call promise chaining.
* If I pass a result of one promise to another promise to another promise resulting in using 3 .then() calls I will end up having 1 .catch() call/calls to handle the errors if they arise.
* If I make an API request and I am returned an array of todos but I only want the fourth item in the array I could write response.data.todos [3].
* The real power of being able to pass one promise to another promise and so on is that you are able to then handle asynchronous requests in a synchronous manner.
* When code is run synchronously that means it stops other code regardless of a time penalty or reduction.
* Async functions make use of generator functions which consume the function by calling the next method and executes until it encounters the yield keyword.
* The await keyword pauses the execution of the remainder of the async function until that asynchronous function call completes.
* The async keyword is a labeller meaning that a specific function will then have some sort of asynchronous action.
* You call the async keyword outside of your function and then use the await keyword inside your function to capture the resolved result.
* Since the async/await method doesn’t have a natural way to raise errors we can utilize a try- catch block inside our async functions to run the desired code and an error be raised.
* When using an async/await function it is common to have multiple data requests scoped locally inside your control statement. This can cause problems when raising errors so it is possible to have an outside function that uses the data if you don’t want it locally scoped.
* The promise.all() method returns a single promise the resolves when every one of the promises passed as an iterable has resolved.
* To open up your Chrome tools to help with development you can use shortcut option-command-i or navigate to the top right hand corner of your browser, click to open all the browser options, navigate down to more tools and then select developer tools.
* The tab on the Chrome Dev Tools that helps with monitoring requests is the network tab.
* If we want to see specifically the returned data when we inspect an xhr request in our Chrome Dev Tools we click on the desired request and then navigate to the preview tab.