**IS312 – Web Design and Programming**

**Student Name:** Trevor Sparks

**Quarter:** Fall 2022

**PE09 Lab Report**

Please complete this document per the lab instructions.

**Task #1: Chapter 11 Exercise: Tracking the Scalpel**

**TS Source Code:**

async function locateScalpel(nest) {

  console.log(nest.name)

  current\_name = nest.name;

  console.log(current\_name);

  for(let i =0; i<100; i++){

    let next\_name = await anyStorage(nest, current\_name, 'scalple')

    if(current\_name == next\_name){

      return current\_name;

    }

    current\_name = next\_name;

  }

}

async function locateScalpel2(nest){

    console.log(nest);

    return new Promise((resolve, reject) =>{

*// trying to find the scalpel*

        let found  = false;

        while(!found){

            anyStorage(nest, nest.name, 'scalpel')

                .then(next\_nest\_name =>{

                    if(next\_nest\_name == nest.name){

                        found = true;

                        resolve("found it!");

                    }

                })

                .catch(err => {

                    reject("did not find it");

                })

        }

    })

}

**Questions:**

1. **Explain the code you wrote for this exercise:**
   1. The idea behind the code I wrote for this exercise is that locateScalpel is an async await function that searches for the scalpel by using the anyStorage promise. The locateScalpel2 function uses promises to search for the scalpel. However, neither of these pieces of code are actually working.
2. **Do request failures properly show up as rejections of the returned promise in both versions?**
   1. Nothing properly showed up because there are mistakes in my code.
3. **If they do, how? If not, why not?**
   1. I believe that I am making errors on both pieces of code. I am not quite sure what I am doing wrong.

**Task #2: Chapter 11 Exercise: Building Promise.ALL**

**TS Source Code:**

let p1 = new Promise((resolve, reject) => {

    let v = Math.random();

    if(v >= .25){

        resolve("Success")

    }else{

        reject("Failed")

    }

})

let p2 = new Promise((resolve, reject) => {

    let v = Math.random();

    if(v >= .25){

        resolve("Success")

    }else{

        reject("Failed")

    }

})

let p3 = new Promise((resolve, reject) => {

    let v = Math.random();

    if(v >= .25){

        resolve("Success")

    }else{

        reject("Failed")

    }

})

let arrayOfPromises = [p1, p2, p3];

function promise\_all(arrayOfPromises){

    let returnValues = [];

    for (let index = 0; index < arrayOfPromises.length; ++index) {

        const element = arrayOfPromises[index];

        elementValue = null;

        element.then((message) =>{

            returnValues.push(message);

        }).catch(() =>{

            return null;

        })

    }

}

promise\_all(arrayOfPromises);

**Questions:**

1. **Explain the code you wrote for this exercise:**
   1. The idea behind this code is that there is an array of promises, and a function that accepts that array of promises as an input. The goal was to have each successful promise push there message to an array and then return that array when the function was finished running. And if a promise did not complete successfully then to not add anything to successful result array. My idea was then to test the length of that array, if it was the same length as the input in array, then the function would return the array. If the array that the promises pushed their messages to was not as long as the input array, then the function would not return anything. That was the idea, however I was not sure how to implement it.
2. **What are the benefits of the promise system?** 
   1. The benefits of using the promise system are that it is easy to catch errors.
3. **Are there any downsides to using promises in asynchronous programming?**
   1. The downsides of promises are that they are really clunky and difficult to read. Looking through another programmers codes with a bunch of .thens (if they are chaining promises) and .catch is a real pain.

**Task #3: Analysis and Rewrite.**

**TS Source Code:**

console.log("The page was loaded");

const url =

  "https://classdemo.amerabyte.net/Sample\_Code/BIT\_285-286-Summer2021/AJAX/process.php";

const nameEL = document.getElementById("name");

const emailEL = document.getElementById("email");

const shaEL = document.getElementById("superheroAlias");

const formEL = document.getElementById("form");

async function onSubmit(event) {

  event.preventDefault();

  const formData = new FormData();

  formData.append("name", name.value);

  formData.append("email", email.value);

  formData.append("superheroAlias", superheroAlias.vale);

  const response = await fetch(url, {

    method: "POST",

    body: formData,

  });

  const jsonData = await response.json();

  console.log(jsonData);

  if (jsonData.success === false) {

    alert(JSON.stringify(jsonData.errors.email));

  } else if (jsonData.success === true) {

    alert("success");

  } else {

    console.log("something else happend");

  }

}

formEL.addEventListener("submit", onSubmit);

**Questions:**

1. **What was the most challenging thing for this exercise?** 
   1. The most challenging part of this exercise for me was simply understanding what was going on. This was the first time that I have ever seen post/fetch and all of the other ways that you interact with an API. Also, when I first tried to make requests I started getting CORS errors. I googled how I should fix this problem, because I thought that it was a problem with my code originally. To get around this I found a chrome extension that allows me to turn of CORS (I am not really sure what that means) but when I made the request I started getting new errors. These new errors were because I was sending JSON object to the server and that was not working. I fixed this problem by submitting the form.
2. **Did you note the use of JavaScript Object Notation (JSON) in building the request to the server in the original code?** 
   1. Yes! When looking at the original code I noticed that I would receive a JavaScript object. This lead me to thinking I should also send JSON data to the php script, but that was not correct. At least I do not think.
3. **Were you able to successfully send and receive the code to/from the server?** 
   1. Yes!!!!!! I was able to successfully send and retrieve data from the server.
4. **This is most used for processing forms, error handling of user input, and for updating pages. Where else do you think it would be useful?**
   1. This has been my favorite assignment then we have done so far this semester. Learning how to actually submit forms is such a real world application of all of what we have been learning. It was really cool!
   2. Retrieving and writing data from a document/server is very useful for more than just processing forms, etc. If you have a database that is used to hold data for data analysis, being able to send requests to a server asynchronously is very useful.

Once you have completed this document please submit per instructions.