**IS312 – Web Design and Programming**

**Student Name:** Trevor Sparks

**Quarter:** Fall 2022

**PE08 Lab Report**

Please complete this document per the lab instructions.

**Task #1: Chapter 8 Exercise: Retry**

**Source Code:**

function Calculate(number1, number2) {

  let result = 0;

  for (;;) {

    try {

      result = primitiveMultiply(number1, number2);

      return result;

    } catch (e) {

      console.log("this try failed, attempting again");

    }

  }

}

**Questions:**

**1)** It allows us to tailor the errors to our specific use case.

**2)** A loop allows us to repeat some code statements given a true condition. The benefits of using recursion is that it is easy to use because it mirrors natural language.

**3)** As long as all calls of primitive multiply are guaranteed to succussed, then no error will be thrown.

**4)** I struggled with this problem. To fix my confusion I looked a couple of pages back in the textbook and was able to find an example that was somewhat similar to this problem.

**Task #2: Chapter 8 Exercise: The Locked Box**

**Source Code:**

const box = {

  locked: true,

  unlock() {

    this.locked = false;

  },

  lock() {

    this.locked = true;

  },

  \_content: [],

  get content() {

    if (this.locked) throw new Error("Locked!");

    return this.\_content;

  },

};

function withBoxUnclocked(passedFunction) {

  try {

    box.unlock();

    passedFunction();

  } catch (error) {

  } finally {

    box.lock();

  }

}

**Questions:**

**1)** It will set locked to true, but because locked is already true.

**2)** An error will be thrown because we are calling content() and box.locked is already true.

**3)** Yes, as long as you call box.unlock() before you access the content property an error will not be thrown.

**4)** The part of this exercise that I found the most interesting was seeing a way that objects, try…catch, and errors could be used in a real world (although rather contrived) example. The part of this exercise that cased me the most difficulty was just understanding what was being asked. I overcame these challenges by working through the problem.

**Task #3: Use Try…Catch & Throw.**

**Source Code:**

*// Upper and lower limit set*

var upperLimit = 21;

var lowerLimit = 0;

*// Initialization of userInput var*

var userInput;

*// function to validate user input. It checks to see if the users input is in between two*

*// pre-specified values*

function myFunction() {

    var i = true;

    while (true) {

        try {

*//getUserInput();*

            userInput = Number(prompt('Please input a number between 1 and 20'));

*// call the validate user function to make sure the user input a valid number*

            if (userInput > lowerLimit && userInput < upperLimit) {

                return userInput;

            }

            else {

                alert("Number not within range, please try again.");

                throw "Number not within range.";

            }

        }

        catch (e) {

            console.log(e);

            continue;

        }

        finally {

            console.log("The user input: " + userInput);

        }

    }

}

**Questions:**

**1)** The most surprising result I found when writing and testing the code was how simple and concise the code became. I could also see how a function like this could be used in a real world setting, weather it was to get a particular number from a user or validating the format of an email/other login details.

**2)** There were a couple of times I forgot to break out of my infinite while loop and the website would not let me reload. I had to exit the page, go into to VSCode and restart the live server, that was kind of a pain, but it all worked out in the end.

**3)** At first I was unable to overcome this obstacle because I did not understand where I was making the mistake. I had to go and read the documentation and find examples of try….catch….finally to finally be able to see where I was making a mistake. Documentation is a life saver.

**4)** I believe that using exception handling in this situation gives us greater flexibility to expand the code that we have written. We could throw and catch errors for the particular case that the number was too large, or that the number was too small. That would be more difficult with the code that was written in PE05.

Once you have completed this document please submit per instructions.