Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**07**

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | **Write a javascript function that takes length, width, and height values of rectangle from user. The function should find the volume of rectangle using the function-currying** |
| 2 | **Create two functions “Profile” (outer function) and “greetingMsg” (inner function) that aims to implements the concept of Function-closure by generating an alert of greeting message with the help of user details provided in outer function.** |
| 3 | **It's a general concept in mathematics where you combine two or more functions into a brand-new function. Write a javascript program to implement the given concept with the help of function-compose for the given function. f(g(x))** |
| 4 | **Write a JavaScript program that uses  filter() to create a filtered array that has all elements with values less than 10 removed.** |
| 5 | **Creates an array consisting of only those elements that satisfy the condition checked by isPositive() function with the help of appropriate JavaScript advance loops concept.** |
| 6 | **Write a JavaScript program that implements the array. Map() that aim to produces an array containing square roots of the numbers in the original array.** |
| 7 | **Create a class named 'Member' having the members: Name, Age, Salary. It also has a method named 'print Salary' which prints the salary of the members. Create child class 'Employee' that inherits the 'Member' class. The 'Employee' classes have data member ‘department'. Now, assign name, age and salary to an employee by making an object of child class and print the same.** |
| 8 | **Write a javascript program to implement the concept of nullish coalesing operator by using the below object properties.** |
| 9 | **Write a javascript program to create the Promise that resolve in 10 seconds and check the status by returning the “Promise is resolved successfully” string if the number is even otherwise reject the promise by returning the string “Promise is rejected”. Convert this task unto async await as well and compare the results.** |

**Task 1: Write a JavaScript function that takes length, width, and height values of rectangle from user. The function should find the volume of rectangle using the function-currying**

**Solution:**

**Index.html**

<nav>

            <ul class="navi" style="list-style: none;display: flex;justify-content: space-around;">

                <a href="task1.html" style="text-decoration:none">

                    <li>Function Currying</li>

                </a>

                <a href="task2.html" style="text-decoration:none">

                    <li>Function-closure</li>

                </a>

                <a href="task3.html" style="text-decoration:none">

                    <li>Function-compose</li>

                </a>

                <a href="task4.html" style="text-decoration:none">

                    <li>Filter() Method</li>

                </a>

            </ul>

            <ul class="navi" style="list-style: none;display: flex;justify-content: space-around;">

                <a href="task5.html" style="text-decoration:none">

                    <li>Advance loops </li>

                </a>

                <a href="task6.html" style="text-decoration:none">

                    <li>Implements Array.map()</li>

                </a>

                <a href="task7.html" style="text-decoration:none">

                    <li>Class Name 'Member' </li>

                </a>

                <a href="task8.html" style="text-decoration:none">

                    <li>Concept of Nullish Coalesing Operator</li>

                </a>

                <a href="task9.html" style="text-decoration:none">

                    <li>Promise [ async await ]</li>

                </a>

            </ul></nav>  
  
**Task1.html**

 <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Calculate Volume Of Rectangle Using Function Curring

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Length</b></label>

                        <input type="number" id="input" class="same" placeholder="Enter length Number">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Width</b></label>

                        <input type="number" id="input2" class="same" placeholder="Enter Width Number">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Height</b></label>

                        <input type="number" id="input3" class="same" placeholder="Enter Height Number">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Volume</b></label>

                        <input type="number" id="input4" class="same" placeholder="Result Of Volume">

                        <button id="btn" class="same" onclick="volumeOfRectangle()">Calculate</button>

                    </div></div></div></section>

**Task1.js**

var length=document.getElementById('input');

var width=document.getElementById('input2');

var heigth=document.getElementById('input3');

var volume=document.getElementById('input4');

    function volumeOfRectangle()

    {

    if(length.value.length>0 && width.value.length>0 && heigth.value.length>0)

    {

        volume.value=rectangle(length.value)(width.value)(heigth.value) // 6

    }

    length.value="";

    width.value="";

    heigth.value="";

}

function rectangle(x) {

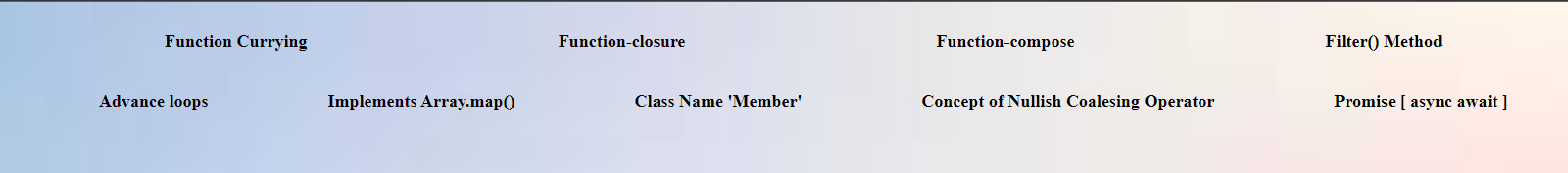
    return (y) => {

        return (z) => {

            return x \* y \* z

        }}}

**Output:**

****

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Task 2: Create two functions “Profile” (outer function) and “greetingMsg” (inner function) that aims to implements the concept of Function-closure by generating an alert of greeting message with the help of user details provided in outer function.**

**Solution:**

**Task2.html**

 <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Concept of Function-Closure

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Name</b></label>

                        <input type="text" id="input" class="same" placeholder="Enter Name....">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Age</b></label>

                        <input type="number" id="input2" class="same" placeholder="Enter Age ...">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Salary</b></label>

                        <input type="number" id="input3" class="same" placeholder="Enter Salary ... ">

                        <button id="btn" class="same" onclick="functionclosure()">Calculate</button>

                    </div> </div></div></section>

**Task2.js**

var name1=document.getElementById('input');

var age=document.getElementById('input2');

var salary=document.getElementById('input3');

function functionclosure()

{

    if(name1.value.length>0 && age.value.length>0 && salary.value.length>0)

    {

        var innerFunc = Profile(name1.value,age.value,salary.value);

        innerFunc();

    }

    length.value="";

    width.value="";

    heigth.value="";

}

function Profile(name,age,salary) {

function greetingMsg() {

    alert(`Your Name Is ${name},Your Age Is ${age},Your Salary Is ${salary}`);

}

return greetingMsg;

}

**Output:**

**Graphical user interface, application

Description automatically generated**

**Text

Description automatically generated**

**Task 3: It's a general concept in mathematics where you combine two or more functions into a brand-new function. Write a javascript program to implement the given concept with the help of function-compose for the given function. f(g(x))**

**Solution:**

**Task3.html**

 <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            function-compose [f(g(x))]

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Number</b></label>

                        <input type="number" id="input" class="same" placeholder="Enter  Number">

                        <label for="" class="same" style="color: white;font-Size: larger"><b>Result</b></label>

                        <input type="number" id="input4" class="same" placeholder="Result Of Volume">

                        <button id="btn" class="same" onclick="check()">Check</button>

                    </div>

                  </div></div></section>

**Task3.js**

var inp=document.getElementById('input');

var result=document.getElementById('input4');

function check()

{

    if(inp.value.length>0)

    {

        const compose = (f,g) => a => f(g(a));

        const g = (x) => x + 15

        const f = (x) => x / 2

        const fg = compose(f,g)

        result.value=fg(inp.value);

        inp.value="";

    }

}

**Output:**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Task 4: Write a JavaScript program that uses  filter() to create a filtered array that has all elements with values less than 10 removed.**

**Solution:**

**Task4.html**

  <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Use Filter Method()

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger;margin-top: 30px"><b>Array [1, 5, 7, 9, 10, 15, 22, 65, 120]</b></label>

                        <input style="margin-top: 30px" type="text" id="input" class="same" placeholder="Number Less than 10">

                        <button id="btn" class="same" onclick="cases()">Enter</button>

                    </div>

                </div>

            </div>

        </section>

**Task4.js**

var result=document.getElementById('input');

function cases(){

const number = [1,5,7,9,10,15,22,65,120];

result.value = number.filter(check);

}

function check(number) {

  return number >= 10;

}

**Output:**

**Graphical user interface, application

Description automatically generated**

**Graphical user interface, application

Description automatically generated**

**Task 5: Creates an array consisting of only those elements that satisfy the condition checked by isPositive() function with the help of appropriate JavaScript advance loops concept.**

**Solution:**

**Task5.html**

  <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            is Positive() function

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger;margin-top: 30px"><b>Array [1, -5, 7, 9, -10, 15, 22, -65, -120]</b></label>

                        <input style="margin-top: 30px" type="text" id="input" class="same" placeholder="Only Positive Number" readonly>

                        <button id="btn" class="same" onclick="check()">Check</button>

                    </div>

                </div>

            </div>

        </section>

**Task5.js**

var result=document.getElementById('input');

function check(){

    const numbers = [1,-5,7,9,-10,15,22,-65,-120];

    result.value=numbers.filter(Ispositive);

}

function Ispositive(num) {

return num > 0;

}

**Output:**

**Graphical user interface, application

Description automatically generated**

**Task 6: Write a JavaScript program that implements the array. Map() that aim to produces an array containing square roots of the numbers in the original array.**

**Solution:**

**Task6.html**

<section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Use Map Method()

                        </h1>

                    </div>

                    <div class="children">

                        <label for="" class="same" style="color: white;font-Size: larger;margin-top: 30px"><b>Array [2, 5, 6, 3, 8, 9]</b></label>

                        <!-- <input type="text" id="input" class="same" placeholder="Number Less than 10"> -->

                        <textarea style="margin-top: 30px" name="" id="input" class="same" cols="30" rows="10" placeholder="Number Less than 10"></textarea>

                        <button id="btn" class="same" onclick="cases()">Enter</button>

                    </div>

                </div>

            </div>

        </section>

**Task6.js**

var result=document.getElementById('input');

function cases(){

    result.value="";

        var arr = [2, 5, 6, 3, 8, 9];

        var newArr = arr.map(function(val, index){

            return {key:index, value:Math.sqrt(val)};

        })

        newArr.forEach(element => {

            result.value = result.value + `${element.value}, `

        });

    }

**Output:**

**Graphical user interface

Description automatically generated**

**Text

Description automatically generated**

**Task 7: Create a class named 'Member' having the members: Name, Age, Salary. It also has a method named 'print Salary' which prints the salary of the members. Create child class 'Employee' that inherits the 'Member' class. The 'Employee' classes have data member ‘department'. Now, assign name, age and salary to an employee by making an object of child class and print the same.**

**Solution:**

**Task7.html**

 <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Class named 'Member' and 'Employee'

                        </h1>

                    </div>

                    <div class="children">

                        <textarea name="text" id="input" class="same" id="" cols="30" rows="10" readonly>

                        </textarea>

                    </div>

                </div>

            </div>

        </section>

**Task7.js**

var txtarea = document.getElementById('input');

class Member {

    constructor(name, age, salary) {

        this.name = name;

        this.age = age;

        this.salary = salary;

    }

}

class Employee extends Member {

    constructor(name, age, salary, department) {

        super(name, age, salary);

        this.department = department;

    }

    printEmpDetails()

    { txtarea.value=`Employee Details \n\nName: ${this.name} \nAge: ${this.age} \nSalary: ${this.salary} \nDepartment: ${this.department}`

     }

}

const a = new Employee('M Muaz Shahzad', 21, 500000, 'BSE 5B');

a.printEmpDetails();

**Output:**

**Text, letter

Description automatically generated**

**Task 8: Write a javascript program to implement the concept of nullish coalesing operator by using the below object properties.**

**Solution:**

**Task8.html**

   <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Nullish Coalesing Operator

                        </h1>

                    </div>

                    <div class="children">

                        <textarea name="text" id="input" class="same" id="" cols="30" rows="5" readonly>

                        </textarea>

                        <button id="btn" class="same" onclick="check()">Check</button>

                    </div>

                </div>

            </div>

        </section>

**Task8.js**

const response = {

  data: {

  name: 'Ronaldo',

  occupation: null,

  lies: 0

  }

}

var txtarea=document.getElementById('input');

var res1=response?.data?.occupation || 'Null ';

var res2=response?.data?.occupation ?? 'Null ';

var res3=response?.data?.lies || '0 Lies';

var res4=response?.data?.lies ?? '0 Lies';

function check(){

  txtarea.value=`Occupation 1: ${res1} \n`

  txtarea.value=txtarea.value+`\nOccupation 2: ${res2} \n`;

  txtarea.value=txtarea.value+`\nlies 1: ${res3} \n`;

  txtarea.value=txtarea.value+`\nlies 2: ${res4} \n`;

}

**Output:**

**Graphical user interface, text, application

Description automatically generated**

**Task 9: Write a javascript program to create the Promise that resolve in 10 seconds and check the status by returning the “Promise is resolved successfully” string if the number is even otherwise reject the promise by returning the string “Promise is rejected”. Convert this task unto async await as well and compare the results.**

**Solution:**

**Task9.html**

  <section>

            <div class="main">

                <div class="child">

                    <div class="children">

                        <h1 class="heading">

                            Promise Without [ async await  ]  5sec

                        </h1>

                    </div>

                    <div class="children">

                        <input type="text" id="input" class="same" placeholder="Enter Value">

                        <input type="button" id="btn" class="same" value="Check" onclick="check()">

                        <textarea name="text" id="input2" class="same" id="" cols="30" rows="5" readonly>

                        </textarea>

                    </div>

                    <div class="children">

                        <h1 class="heading">

                            Promise  [ async await  ]    5sec

                        </h1>

                        <input type="text" id="input3" class="same" placeholder="Enter Value">

                        <input type="button" id="btn" class="same" value="Check" onclick="check1()">

                        <textarea name="text" id="input4" class="same" id="" cols="30" rows="5" readonly>

                        </textarea>

                    </div>

                </div>

            </div>

        </section>

**Task9.js**

      var text1=document.getElementById('input');

      var txtarea=document.getElementById('input2');

      var text2=document.getElementById('input3');

      var txtarea2=document.getElementById('input4');

      function myDisplayer(some) {

        txtarea.value=some;

      }

      function check(){

          if(text1.value.length>0)

          {

            let myPromise = new Promise(function(myResolve, myReject) {

                if (text1.value % 2 !== 0) {

                myResolve("Promise is resolved successfully");

                } else {

                  myReject("Promise is rejected");

                }

              });

              myPromise.then(

                function(value) {myDisplayer(value);},

                function(error) {myDisplayer(error);}

              );

          }

      }

      function myDisplayer2(some) {

        txtarea2.value=some;

      }

      async  function check1(){

        console.log("SS")

        if(text2.value.length>0)

        {

          console.log("SSs")

          let myPromise = new Promise(function(myResolve, myReject) {

            let x=0;

              if (text2.value % 2 !== 0) {

                  console.log("SSsss")

                  setTimeout(function() {myResolve("Promise is resolved successfully");}, 5000);

              } else {

                myReject("Promise is rejected");

              }

              console.log("SSz")

            });

            await   myPromise.then(

              function(value) {myDisplayer2(value);},

              function(error) {myDisplayer2(error);}

            );

        }

    }

**Output:**

**Graphical user interface, application

Description automatically generated**