**ASSIGNMENT**Submit your solutions here - <https://forms.gle/yRhahFJpnTGjtZLQ6>

1. Write a JavaScript program that displays the largest integer among two integers.

function findLargestInteger(a, b) {

if (a > b) {

return a;

} else {

return b;

}

}

var num1 = 10;

var num2 = 5;

var largestInteger = findLargestInteger(num1, num2);

console.log("The largest integer is: " + largestInteger);

1. Write a JavaScript function to find the sign of the product of three numbers. Display an alert box with the specified sign.   
   Sample numbers : 3, -7, 2  
   Output : The sign is -

function findProductSign(num1, num2, num3) {

var product = num1 \* num2 \* num3;

if (product > 0) {

alert("The sign is +");

} else if (product < 0) {

alert("The sign is -");

} else {

alert("The product is zero.");

}

}

var n1 = 3;

var n2 = -7;

var n3 = 2;

findProductSign(n1, n2, n3);

1. Write a JavaScript function to find the largest of five numbers. Display an alert box to show the results.   
   Sample numbers : -5, -2, -6, 0, -1   
   Output : 0

function findLargestNumber(num1, num2, num3, num4, num5) {

var largest = Math.max(num1, num2, num3, num4, num5);

alert("The largest number is: " + largest);

}

var number1 = -5;

var number2 = -2;

var number3 = -6;

var number4 = 0;

var number5 = -1;

findLargestNumber(number1, number2, number3, number4, number5);

1. Write a JavaScript program that iterates integers from 1 to 100. But for multiples of three print "Fizz" instead of the number and for multiples of five print "Buzz". For numbers multiples of both three and five print "FizzBuzz".

for (var i = 1; i <= 100; i++) {

if (i % 3 === 0 && i % 5 === 0) {

console.log("FizzBuzz");

} else if (i % 3 === 0) {

console.log("Fizz");

} else if (i % 5 === 0) {

console.log("Buzz");

} else {

console.log(i);

}

}

1. According to Wikipedia a happy number is defined by the following process : "Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers (or sad numbers)". Write a JavaScript program to find and print the first 5 happy numbers.

function isHappyNumber(num) {

var visited = {};

while (num !== 1 && !visited[num]) {

visited[num] = true;

var sum = 0;

var digits = num.toString().split('');

for (var i = 0; i < digits.length; i++) {

sum += Math.pow(parseInt(digits[i]), 2);

}

num = sum;

}

return num === 1;

}

function findHappyNumbers(count) {

var happyNumbers = [];

var number = 1;

while (happyNumbers.length < count) {

if (isHappyNumber(number)) {

happyNumbers.push(number);

}

number++;

}

return happyNumbers;

}

var firstFiveHappyNumbers = findHappyNumbers(5);

console.log("The first 5 happy numbers are: " + firstFiveHappyNumbers.join(', '));

1. Write a JavaScript program to find the Armstrong numbers of 3 digits. Note : An Armstrong number of three digits is an integer such that the sum of the cubes of its digits is equal to the number itself. For example, 371 is an Armstrong number since 3\*\*3 + 7\*\*3 + 1\*\*3 = 371.

function isArmstrongNumber(num) {

var sum = 0;

var temp = num;

var numDigits = 3;

while (temp > 0) {

var digit = temp % 10;

sum += Math.pow(digit, numDigits);

temp = Math.floor(temp / 10);

}

return num === sum;

}

function findArmstrongNumbers() {

var armstrongNumbers = [];

for (var i = 100; i < 1000; i++) {

if (isArmstrongNumber(i)) {

armstrongNumbers.push(i);

}

}

return armstrongNumbers;

}

var armstrongNumbers = findArmstrongNumbers();

console.log("Armstrong numbers of 3 digits: " + armstrongNumbers.join(', '));

1. Write a JavaScript program to sum 3 and 5 multiples under 1000.

function sumMultiples() {

var sum = 0;

for (var i = 1; i < 1000; i++) {

if (i % 3 === 0 || i % 5 === 0) {

sum += i;

}

}

return sum;

}

var result = sumMultiples();

console.log("The sum of multiples of 3 and 5 below 1000 is: " + result);

1. Write a node js script using file system which will   
   - create a new file   
   - add data to a file  
   - append data to a file  
   - read an html file and display it on the browser

const fs = require('fs');

const http = require('http');

fs.writeFile('example.txt', 'This is some example data.', (err) => {

if (err) throw err;

console.log('File created and data added.');

fs.appendFile('example.txt', '\nThis is additional data.', (err) => {

if (err) throw err;

console.log('Data appended to the file.');

fs.readFile('index.html', 'utf8', (err, data) => {

if (err) throw err;

const server = http.createServer((req, res) => {

res.writeHead(200, { 'Content-Type': 'text/html' });

res.end(data);

});

server.listen(3000, () => {

console.log('Server is running. Open your browser and visit http://localhost:3000');

});

});

});

});

1. What is a URL. Break it down into several parts that the url consists of. Use URL module to write a nodejs script to show the breakdown parts of a url as well.

const url = require('url');

const myUrl = '<https://www.example.com:8080/products?page=1&sort=asc#section-1>';

const parsedUrl = url.parse(myUrl);

const protocol = parsedUrl.protocol;

const hostname = parsedUrl.hostname;

const port = parsedUrl.port;

const path = parsedUrl.pathname;

const query = parsedUrl.query;

const fragment = parsedUrl.hash;

console.log('Protocol:', protocol);

console.log('Hostname:', hostname);

console.log('Port:', port);

console.log('Path:', path);

console.log('Query:', query);

console.log('Fragment:', fragment);

1. What is a Promise in JavaScript and how does it work?

Promise in js is an eventual result of an asynchronous operation. It is a way to handle asynchronous tasks and manage their outcomes more easily. Imagine you have an asynchronous task, such as making an API request or reading a file from the disk. Normally, JavaScript would continue executing the code while waiting for the task to complete. However, you may want to perform some actions only when the task has finished and the result is available. That's where Promises come in. They provide a structured way to handle asynchronous operations and their results.