**1. JS-FUNCTION-P-LEAPYEAR-1 (2 points)**

Write the function isLeapYear according to ES5 standard. **Use a function declaration** .

The function takes a year to be checked as a parameter.

The function Returns true if the given parameter value is a leap year and false if it is not a leap year.

It should be noted that leap years must meet the following two conditions at the same time: 1) when the year is divided by four, the remainder is zero, and 2) (when the year is divided by one hundred, the remainder is not zero) or (when the year is divided by four hundred, the remainder is zero).

Fill a year into a textbox on a web page. Call the function by clicking the button Check year and display (by utilizing another function) "Year xxxx is a leap year" or "Year xxxx is not a leap year" in a div below. The xxx is the filled in year.

WE WILL PROGRAM THIS TOGETHER.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Leap Year Checker</title>

</head>

<body>

    <label for="yearInput">Enter a year:</label>

    <input type="text" id="yearInput" placeholder="Enter a year">

    <button onclick="checkLeapYear()">Check year</button>

    <div id="result"></div>

    <script>

        function isLeapYear(year) {

            //leap year conditions

            return (year % 4 === 0 && (year % 100 !== 0 || year % 400 ===0));

        }

        function checkLeapYear() {

            var yearInput = document.getElementById('yearInput').value;

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

            if (!isNaN(yearInput) && yearInput !== '') {

                var year = parseInt(yearInput);

                var isLeap = isLeapYear(year);

                resultDiv.innerHTML = "Year " + (isLeap ? "is a leap year." : "is not a leap year.");

            } else {

                resultDiv.innerHTML = "Please enter a valid year.";

            }

        }

    </script>

</body>

</html>

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**2. JS-ARRAY-P-CONTAINS-NUMBER-1 (1 point)**

Implement the function containsNumber according to ES5 standard. However, **use this time a function expression** .

The function takes two arguments. The first argument is numbers , which is an array of numbers. The second argument is aNumber , which is the number to search.

An example of one possible elements array:

[6,4,2,5,9,7,5,7,2]s

The function Returns true , if the array contains the number given in aNumber . Otherwise the function returns false .

Display the result on the web page as in JS-FUNCTION-P-LEAPYEAR-1. This time the result text should be like “Array contains the number x” or “Array doen't contain the number x”.  
  
<!DOCTYPE html>

<html>

<head>

    <title>Leap Year Checker</title>

</head>

<body>

    <label for="numberInput">Enter a number:</label>

    <input type="text" id="numberInput">

    <button onclick="checkNumber()">Check number</button>

    <div id="result"></div>

    <script>

        var givenArray = [6,4,2,5,9,7,5,7,2]

        var checkNumber = function () {

            var numberInput = document.getElementById('numberInput').value;

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

            var numberToCheck = parseInt(numberInput);

            if (givenArray.indexOf(numberToCheck) !== -1){

                resultDiv.innerHTML = "Array contains the number " +  numberToCheck;

            } else {

                resultDiv.innerHTML = "Array doesn't contain the number " + numberToCheck;

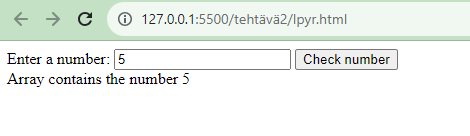
            }

        };

    </script>

</body>

</html>



**3. JS-FUNCTION-P-CONVERT-TO-MINUTE-FORMAT-1 (2 points)**

Write the function convertToMinutesFormat . Use this time an arrow function introduced in ES6.

The function takes a hoursInHundredths as a parameter. The function should be able to handle a parameter value that is given in one of the following formats: x.xx, xx.xx, x,xx or xx,xx.

Example parameter values: 3.40 , 03.20 , 0.15 , 14.80 .

The function Returns hours and minutes in one of the following formats: h:mm or hh:mm.

From the above example parameter values ​​the function returns: 3:24, 3:12, 0:09, 14:48.

Fill an hour time to convert into a textbox on a web page. Call the function by clicking the button “Convert to Minutes” and display the result like (by utilizing another function) "3.20 hours is in hours and minutes equal to 3:12"

Take care of the necessary rounding of minutes.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Hour to Minute</title>

</head>

<body>

    <label for="hourInput">Enter Hour</label>

    <input type="text" id="hourInput">

    <button onclick="convertToMinute()">Convert to Minute</button>

    <div id="result"></div>

    <script>

        const convertToMinutesFormat = (hoursInHundredths) => {

            const hours = Math.floor(hoursInHundredths);

            const minutes = Math.round((hoursInHundredths - hours) \* 60);

            // Add leading zero for single-digit minutes

            const formattedMinutes = (minutes < 10) ? `0${minutes}` : `${minutes}`;

            return `${hours}:${formattedMinutes}`;

        }

        const convertToMinute = () => {

            const hourInput = document.getElementById('hourInput').value;

            const resultDiv = document.getElementById('result');

            // Use parseFloat to convert input value to a floating-point number

            const givenInput = parseFloat(hourInput);

            // Check if the input is a valid number

            if (!isNaN(givenInput)) {

                const minuteOutput = convertToMinutesFormat(givenInput);

                resultDiv.innerHTML = `${givenInput} hours is in hours and minutes equal to ${minuteOutput}`;

            } else {

                resultDiv.innerHTML = 'Please enter a valid number for hours.';

            }

        }

    </script>

</body>

</html>

**4. JS-FUNCTION-T-UNDERSTAND-LIBRARY-FUNCTIONS-1 (1 point)**

Search a good JavaScript reference from the Net. This time we focus on functions handling arrays.

1. What is the url of the reference you found?

https://www.w3schools.com/js/js\_arrow\_function.asp

1. What does it mean that an operation mutates an array in place? (0.5 points)

When an operation mutates an array in place, it means that the original array is modified directly, without creating a new array. The changes happen to the existing array object, and there is no need to assign the result to a new variable.

1. What does it mean then an operation Returns a copy of an array? (0.5 points>

When an operation returns a copy of an array, it means that a new array is created, leaving the original array unchanged. The operation produces a new array with the desired changes, and we need to assign this new array to a variable if we want to use or keep the modified version.

**5. JS-OBJECT-P-BASICS-1 (2 points)**

This time we concentrate on object literals. Write code clips to

1. Create an object book containing the following properties : isbn , name , authors , publicationDate . (0.5 points)

const book = {

isbn: '1234567890',

name: 'Sample Book',

authors: ['Author1', 'Author2'],

publicationDate: '2023-01-01'

};

1. Add the following methods with the following names to the book object: getAuthors , setAuthors, getIsbn, setIsbn . (0.5 points)

const book = {

    isbn: '123456',

    name: 'Sample Book',

    authors: ['Author1', 'Author2'],

    publicationDate: '2023-01-01',

    getAuthors: function(){

        return this.authors;

    },

    setAuthors: function (newAuthors){

        this.authors = newAuthors;

    },

    getIsbn: function(){

        return this.isbn;

    },

    setIsbn: function(newIsbn){

        this.isbn = newIsbn;

    }

};

1. Create two book objects. Compare if they model the same book. You can use the value of the field isbn as comparison criteria in Defining equality: Same isbn value, same book. (0.5 points)

A screenshot of a computer code

Description automatically generated

1. Create two book objects with exactly the same values ​​in all the features. Do they have the same identity? (0.5 points)

A screenshot of a computer program

Description automatically generated

**6. JS-OBJECT-P-CONVERT-OUNCES-TO-GRAMS-1 (2 points)**

Write the function convertOuncesToGrams . You can use ES6 standard features.

The function takes measurements as a parameter.

The measurements parameter is an array containing objects.

An example of the value of the measurements parameter:

[{ batchid: 434, unit: “ounce”, weight: 12.21 }, {batchid: 414, unit: “gram”, weight: 199.54 },{ batchid: 522, unit: “ounce”, weight: 18.88 }]

The function Returns an array where all the measurements are in grams. Like the following:

[{ batchid: 434, unit: “gram”, weight: 346.15 }, {batchid: 414, unit: “gram”, weight: 199.54 },{ batchid: 522, unit: “gram”, weight: 535.24 }]

Please, give the results with two digits.

function convertOuncesToGrams(measurements) {

    const gramsPerOunce = 28.3495; // Conversion factor

    // Initialize an empty array to store the converted measurements

    const convertedMeasurements = [];

    // Iterate through each measurement in the array

    for (let i = 0; i < measurements.length; i++) {

      const measurement = measurements[i];

      // Extract properties from the measurement

      const batchid = measurement.batchid;

      const unit = measurement.unit;

      const weight = measurement.weight;

      // Check if the unit is "ounce" and convert the weight to grams

      let convertedWeight;

      if (unit === "ounce") {

        convertedWeight = (weight \* gramsPerOunce).toFixed(2);

      } else {

        convertedWeight = weight.toFixed(2);

      }

      // Create a new object with converted values

      const convertedMeasurement = {

        batchid: batchid,

        unit: "gram",

        weight: parseFloat(convertedWeight),

      };

      // Push the converted measurement to the array

      convertedMeasurements.push(convertedMeasurement);

    }

    // Return the array of converted measurements

    return convertedMeasurements;

  }

  // Example usage:

  const measurements = [

    { batchid: 434, unit: "ounce", weight: 12.21 },

    { batchid: 414, unit: "gram", weight: 199.54 },

    { batchid: 522, unit: "ounce", weight: 18.88 },

  ];

  const convertedMeasurements = convertOuncesToGrams(measurements);

  console.log(convertedMeasurements);