WEB TECHNOLOGIES ASSIGNMENT 1

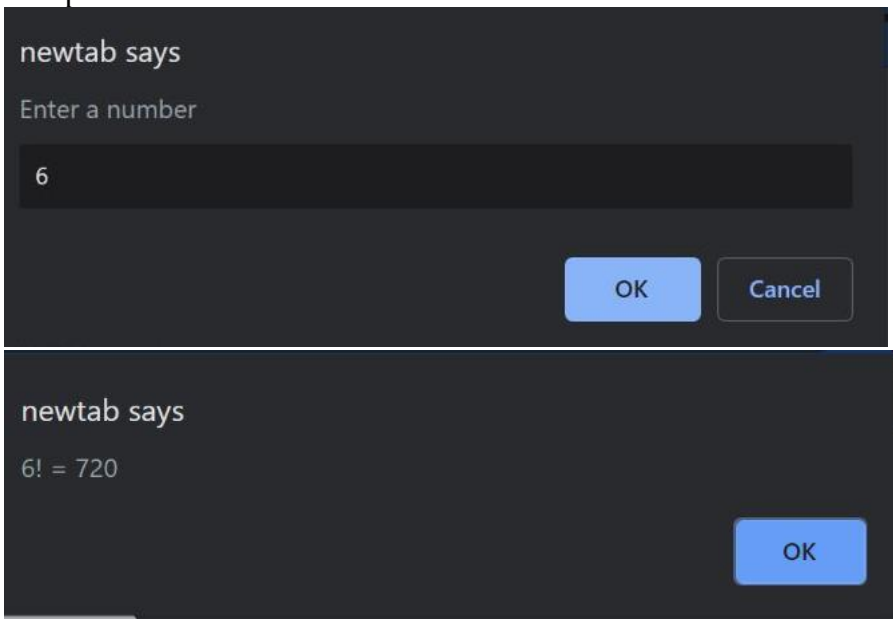
Srividya Krishnakumar
CS6A
55

1. Write a JavaScript program to find the factorial of a number.

```
function factorial(n) {
    if (n<=0) return 1;
    return n*factorial(n-1);
}

const a = Number(prompt("Enter a number"));
if (isNaN(a)) {
    alert(`${a} is not a number!`);
} else {
    const f = factorial(a);
    alert(`${a}! = ${f}`);
}
```

Output:

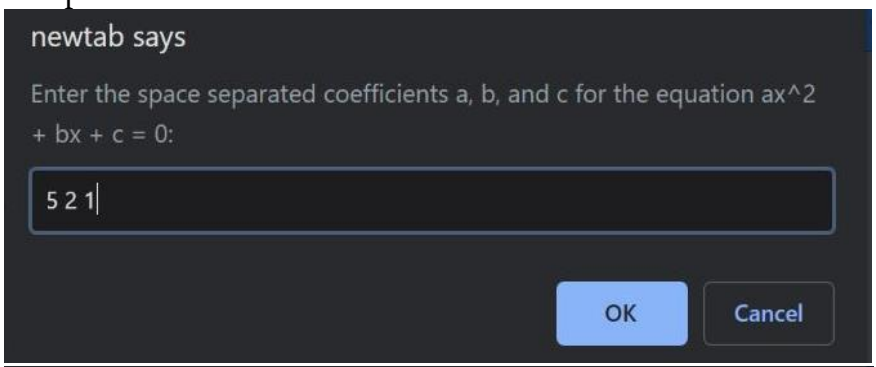


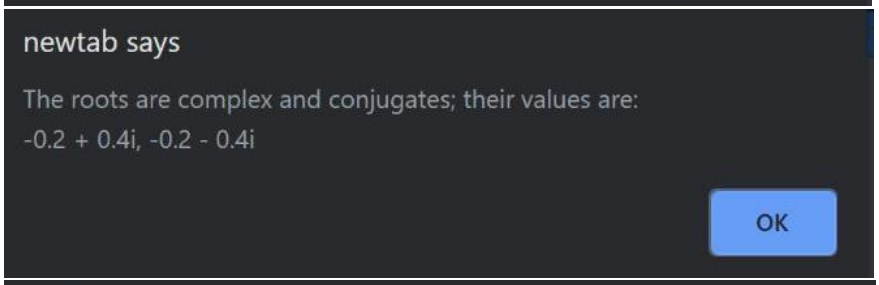
2. Write a JavaScript code to solve a quadratic equation by reading the coefficients through dialog box. Also use confirm dialog box to check whether user wants to continue or not.

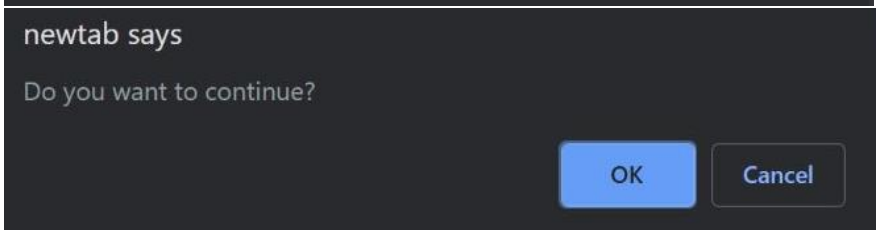
```
do {
    const msg = 'Enter the space separated coefficients a, b, and c for
      the equation ax^2 + bx + c = 0: ';
    const coeffs = prompt(msg).split(' ');
    const a = Number(coeffs[0]), b = Number(coeffs[1]),
          c = Number(coeffs[2]);
    if (!(isNaN(a) || isNaN(b) || isNaN(c) )) {
        const d = b**2 - 4*a*c;
        const r = -b/(2*a);
        if (d>0) { // the roots are real and distinct
            const sqrtD = Math.sqrt(d)/(2*a);
            const r1 = r + sqrtD;
            const r2 = r - sqrtD;
            alert(`The roots are real and distinct; their values are: \n
```

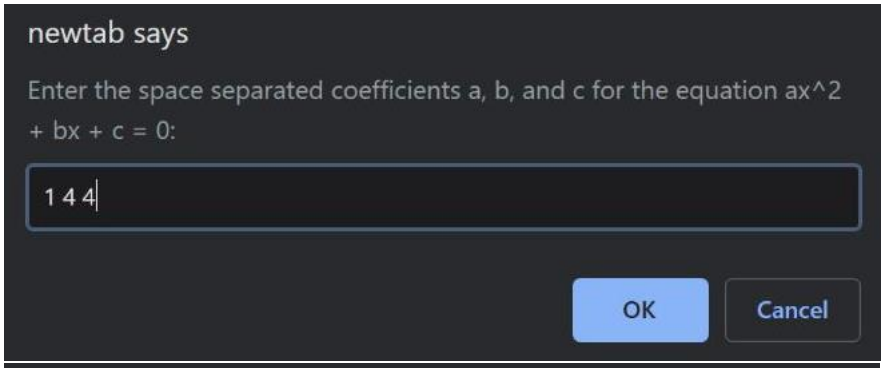
```
            ${r1}, ${r2}`);
        } else if (d===0) { // the roots are real and equal
            alert(`The roots are real and equal; their values are: \n
            ${r}, ${r}`);
        } else { // the root are complex and conjugates of each other
            const sqrtD = Math.sqrt(-d)/(2*a);
            const r1 = `${r} + ${sqrtD}i`;
            const r2 = `${r} - ${sqrtD}i`;
            alert(`The roots are complex and conjugates; their values are:
            \n${r1}, ${r2}`);
        }
    } else {
        alert('Enter only numeric values');
    }
} while (confirm('Do you want to continue?'));
```

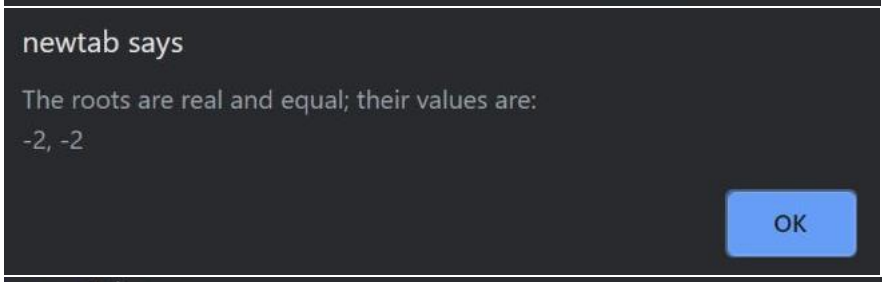
Output:

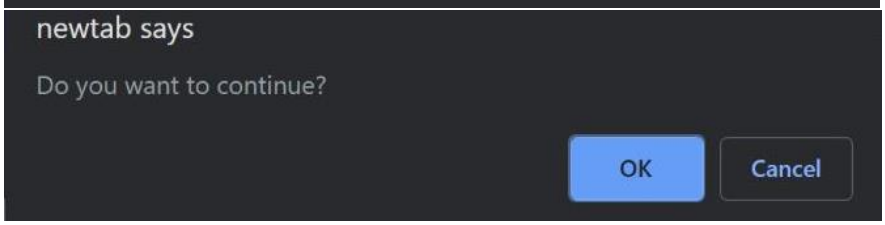


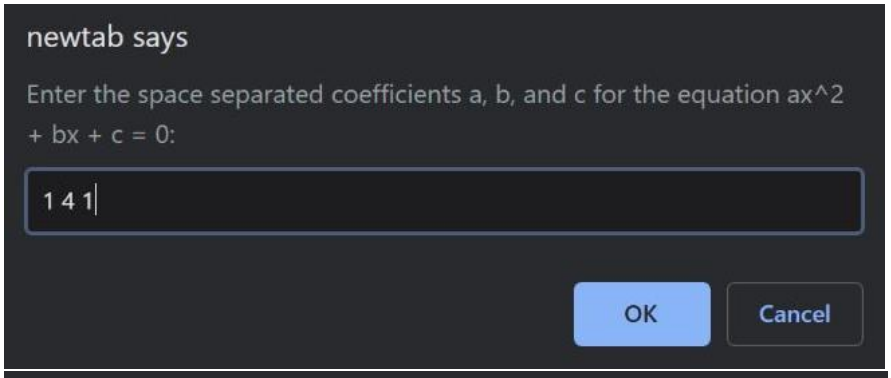


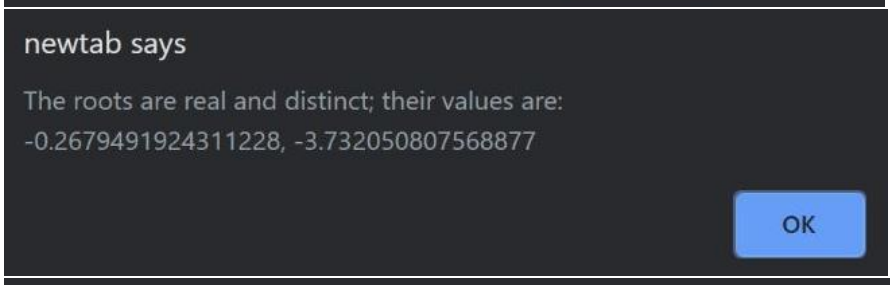


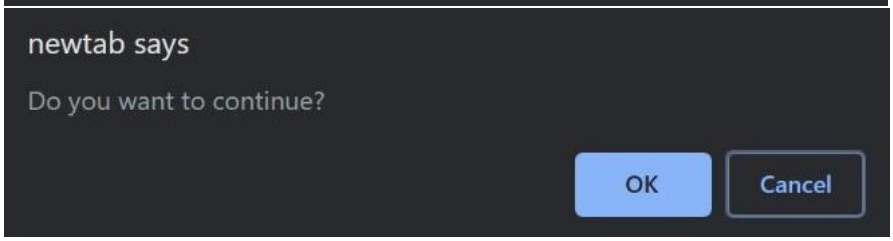

newtab says
Enter the space separated coefficients a, b, and c for the equation ax^2 + bx + c = 0:
1 4 4
OK
Cancel


newtab says
The roots are real and equal; their values are:
-2, -2
OK


newtab says
Do you want to continue?
OK
Cancel


newtab says
Enter the space separated coefficients a, b, and c for the equation ax^2 + bx + c = 0:
1 4 1
OK
Cancel


newtab says
The roots are real and distinct; their values are:
-0.2679491924311228, -3.732050807568877
OK


newtab says
Do you want to continue?
OK
Cancel

3. Explain various String properties and methods present in JavaScript.

String Properties:

Property	Description
constructor	Returns the string's constructor function.
length	Returns the length of the string.
prototype	Allows you to add new properties and methods to a String object.

String Methods:

Method	Description
charAt(position)	Returns the character at the specified position (in Number).
charCodeAt(position)	Returns a number indicating the Unicode value of the character at the given position (in Number).
concat([string,,])	Joins specified string literal values (specify multiple strings separated by comma) and returns a new string.
indexOf(SearchString, Position)	Returns the index of first occurrence of specified String starting from specified number index. Returns -1 if not found.
lastIndexOf(SearchString, Position)	Returns the last occurrence index of specified SearchString, starting from specified position. Returns -1 if not found.
localeCompare(string,position)	Compares two strings in the current locale.
match(RegExp)	Search a string for a match using specified regular expression. Returns a matching array.
replace(searchValue, replaceValue)	Search specified string value and replace with specified replace Value string and return new string. Regular expression can also be used as searchValue.
search(RegExp)	Search for a match based on specified regular expression.
slice(startNumber, endNumber)	Extracts a section of a string based on specified starting and ending index and returns a new string.
split(separatorString, limitNumber)	Splits a String into an array of strings by separating the string into substrings based on specified separator. Regular expression can also be used as separator.
substr(start, length)	Returns the characters in a string from specified starting position through the specified number of characters (length).
substring(start, end)	Returns the characters in a string between start and end indexes.
toLocaleLowerCase()	Converts a string to lower case according to current locale.
toLocaleUpperCase()	Converts a sting to upper case according to current locale.
toLowerCase()	Returns lower case string value.
toString()	Returns the value of String object.
toUpperCase()	Returns upper case string value.
valueOf()	Returns the primitive value of the specified string object.

4. Create a sample form program that collects the first name, last name, email, user id, password and confirms password from the user. All the inputs are mandatory and email address entered should be in correct format. Also, the values entered in the password and confirm password textboxes should be the same. After validating using JavaScript, in output display proper error messages in red color just next to the textbox where there is an error.

index.html

```
<html>
<head>
    <title>Form Validation</title>
    <style>
        span { color: red; }
    </style>
</head>
<body>
    <table id='table1'>
        <tr>
            <td>First Name:</td>
            <td><input type='text' id='first' onkeyup='validate();' /></td>
            <td>
                <div id='errFirst'></div>
            </td>
        </tr>
        <tr>
            <td>Last Name:</td>
            <td><input type='text' id='last' onkeyup='validate();' /></td>
            <td>
                <div id='errLast'></div>
            </td>
        </tr>
        <tr>
            <td>Email:</td>
            <td><input type='text' id='email' onkeyup='validate();' /></td>
            <td>
                <div id='errEmail'></div>
            </td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><input type='password' id='password' onkeyup='validate();'
                   /></td>
            <td>
                <div id='errPassword'></div>
            </td>
        </tr>
        <tr>
            <td>Confirm Password:</td>
            <td><input type='password' id='confirm' onkeyup='validate();'
```

```
                /></td>
            <td>
                <div id='errConfirm'></div>
            </td>
        </tr>
        <tr id='td-submit'>
            <td><input type='button' id='submit' value='Submit'
                  onclick='validate();finalValidate();' /></td>
            <td>
                <div id='errFinal'></div>
            </td>
        </tr>
    </table>

    <script src="script.js"></script>
</body>
</html>
```

script.js

```
let divs = new Array();
divs[0] = 'errFirst';
divs[1] = 'errLast';
divs[2] = 'errEmail';
divs[3] = 'errPassword';
divs[4] = 'errConfirm';
function validate() {
    let inputs = new Array();
    inputs[0] = document.getElementById('first').value;
    inputs[1] = document.getElementById('last').value;
    inputs[2] = document.getElementById('email').value;
    inputs[3] = document.getElementById('password').value;
    inputs[4] = document.getElementById('confirm').value;
    let errors = new Array();
    errors[0] = "<span>Please enter your first name!</span>";
    errors[1] = "<span>Please enter your last name!</span>";
    errors[2] = "<span>Please enter your email!</span>'";
    errors[3] = "<span>Please enter your password!</span>'";
    errors[4] = "<span>Please confirm your password!</span>'";
    for (i in inputs) {
        let errMessage = errors[i];
        let div = divs[i];
        if (inputs[i] === '')
            document.getElementById(div).innerHTML = errMessage;
        else if (i == 2) {
            let atpos = inputs[i].indexOf('@');
            let dotpos = inputs[i].lastIndexOf('.');
            if (atpos < 1 || dotpos < atpos + 2 ||
                  dotpos + 2 >= inputs[i].length)
```

```
                document.getElementById('errEmail').innerHTML =
                  "<span>Enter a valid email address!</span>";
            else
                document.getElementById(div).innerHTML = 'OK!';
        }
        else if (i == 4) {
            let first = document.getElementById('password').value;
            let second = document.getElementById('confirm').value;
            if (second != first)
                document.getElementById('errConfirm').innerHTML =
                  "<span>Your passwords don't match!</span>";
            else
                document.getElementById(div).innerHTML = 'OK!';
        }
        else
            document.getElementById(div).innerHTML = 'OK!';
    }
}
function finalValidate() {
    let count = 0;
    for (i = 0; i < 5; i++) {
        let div = divs[i];
        if (document.getElementById(div).innerHTML == 'OK!')
            count++;
    }
    if (count === 5)
        document.getElementById('errFinal').innerHTML = 'All the data you ente
red is correct!!!';
}
```

Output

First Name: asd OK!
Last Name: Please enter your last name!
Email: asd Enter a valid email address!
Password: ••• OK!
Confirm Password: •••• Your passwords don't match!
Submit

First Name: asdad OK!
Last Name: asda OK!
Email: asd@abc.com OK!
Password: •••• OK!
Confirm Password: •••• OK!
Submit All the data you entered is correct!!!

5. What is === operator in JavaScript?

JavaScript has both strict and type–converting comparisons. A strict comparison (e.g., ===) is only true if the operands are of the same type and the contents match. The more commonly used abstract comparison (e.g. ==) converts the operands to the same type before making the comparison.

Features of comparisons:

- Two strings are strictly equal when they have the same sequence of characters, same length, and same characters in corresponding positions.
- Two numbers are strictly equal when they are numerically equal (have the same number value). NaN is not equal to anything, including NaN. Positive and negative zeros are equal to one another.
- Two Boolean operands are strictly equal if both are true or both are false.
- Two distinct objects are never equal for either strict or abstract comparisons.
- An expression comparing Objects is only true if the operands reference the same Object.
- Null and Undefined Types are strictly equal to themselves and abstractly equal to each other.

6. Syntactic differences between HTML and XHTML.

- XHTML is a combination of HTML and XML (Extensible Markup Language).
- XHTML consists of all the elements in HTML 4.01, combined with the strict syntax of XML.
- XML is a markup language where everything must be marked up correctly, which results in "well-formed" documents.
- XML is designed to describe data, and HTML is designed to display data.

The Most Important Differences from HTML:

1. Document Structure
    - XHTML DOCTYPE is **mandatory.**
    - The xmlns attribute in <html> is **mandatory.**
    - <html>, <head>, <title>, and <body> are **mandatory.**
2. XHTML Elements
    - XHTML elements must be **properly nested.**
    - XHTML elements must always be **closed.**
    - XHTML elements must be in **lowercase.**
    - XHTML documents must have **one root element.**
3. XHTML Attributes
    - Attribute names must be in **lower case.**
    - Attribute values must be **quoted.**
    - Attribute minimization is **forbidden.**
4. The *id* attribute **replaces** the *name* attribute.
5. The XHTML DTD defines **mandatory** elements.

How to Convert from HTML to XHTML

1. Add an XHTML <!DOCTYPE> to the first line of every page.

2. Add an *xmlns* attribute to the html element of every page.
3. Change all element names to lowercase.
4. Close all empty elements.
5. Change all attribute names to lowercase.
6. Quote all attribute values.