Problem statement

Create a calculator using DOM.

Use Keyboard events only for numbers.

When any keys pressed other than number keys show an alert warning "Only numbers are allowed".

Add functions to perform addition, subtraction, division, multiplication and modulus.

The calculator must handle infix expressions.

Use DOM to create html elements

For styles you can use css and bootstrap.

UI should be responsive for all media screens.

Include the test suite in your html file which is given below.

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>DOM Calculator</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<style>

body {

*display*: flex;

*align-items*: center;

*justify-content*: center;

*height*: 100vh;

*background-color*: #f4f4f4;

}

#calculator {

*background*: #e6e6e6;

*border-radius*: 10px;

*box-shadow*: 0 5px 10px rgba(0,0,0,0.2);

*padding*: 20px;

*width*: 300px;

}

#display {

*width*: 100%;

*height*: 60px;

*font-size*: 2rem;

*text-align*: right;

*padding*: 10px;

*margin-bottom*: 10px;

*border*: none;

*background-color*: #ccc;

*border-radius*: 5px;

}

.btn-grid {

*display*: grid;

*grid-template-columns*: repeat(4, 1fr);

*gap*: 10px;

}

button {

*padding*: 15px;

*font-size*: 1.2rem;

*border*: none;

*border-radius*: 5px;

}

.btn-blue {

*background-color*: #007bff;

*color*: white;

}

.btn-red {

*color*: red;

}

.btn-purple {

*color*: purple;

}

</style>

</head>

<body>

<div id="calculator">

<input type="text" id="display" value="0" disabled>

<div class="btn-grid" id="buttons"></div>

</div>

<script src="script.js"></script>

<!-- Test Suite -->

<script>

*function* runTests() {

*const* expression = "5+5\*2";

*const* result = evaluateExpression(expression);

console.assert(result === 15, `Expected 15, but got ${result}`);

*const* expression2 = "20/4+2";

*const* result2 = evaluateExpression(expression2);

console.assert(result2 === 7, `Expected 7, but got ${result2}`);

*const* expression3 = "10%3";

*const* result3 = evaluateExpression(expression3);

console.assert(result3 === 1, `Expected 1, but got ${result3}`);

}

window.onload = runTests;

</script>

</body>

</html>

[script.js](http://script.js)

*const* display = document.getElementById('display');

*const* buttonsContainer = document.getElementById('buttons');

*let* expression = '';

*const* buttons = [

'C', '←', '.', '×',

'7', '8', '9', '÷',

'4', '5', '6', '-',

'1', '2', '3', '+',

'0', '00', '=', '%'

];

// Helper to update display

*function* updateDisplay(*value*) {

display.value = *value* || '0';

}

// DOM: Create buttons

buttons.forEach(*text* *=>* {

*const* btn = document.createElement('button');

btn.textContent = *text*;

btn.className = 'btn ' + (*text* === '=' ? 'btn-blue' :

*text* === 'C' || *text* === '←' ? 'btn-red' :

['×', '÷', '+', '-', '%'].includes(*text*) ? 'btn-purple' : '');

btn.addEventListener('click', () *=>* handleButtonClick(*text*));

buttonsContainer.appendChild(btn);

});

// Handle click

*function* handleButtonClick(*btn*) {

if (*btn* === 'C') {

expression = '';

} else if (*btn* === '←') {

expression = expression.slice(0, -1);

} else if (*btn* === '=') {

try {

expression = evaluateExpression(expression).toString();

} catch {

expression = 'Error';

}

} else {

expression += *btn*;

}

updateDisplay(expression);

}

// Evaluate expression safely

*function* evaluateExpression(*exp*) {

return *Function*('"use strict"; return (' + *exp*.replace(/×/g, '\*').replace(/÷/g, '/') + ')')();

}

// Handle keyboard events

document.addEventListener('keydown', (*e*) *=>* {

if (/[0-9]/.test(*e*.key)) {

expression += *e*.key;

updateDisplay(expression);

} else if (['+', '-', '\*', '/', '%'].includes(*e*.key)) {

expression += *e*.key;

updateDisplay(expression);

} else if (*e*.key === 'Enter') {

expression = evaluateExpression(expression).toString();

updateDisplay(expression);

} else if (*e*.key === 'Backspace') {

expression = expression.slice(0, -1);

updateDisplay(expression);

} else {

alert('Only numbers are allowed');

}

});

Output -   
