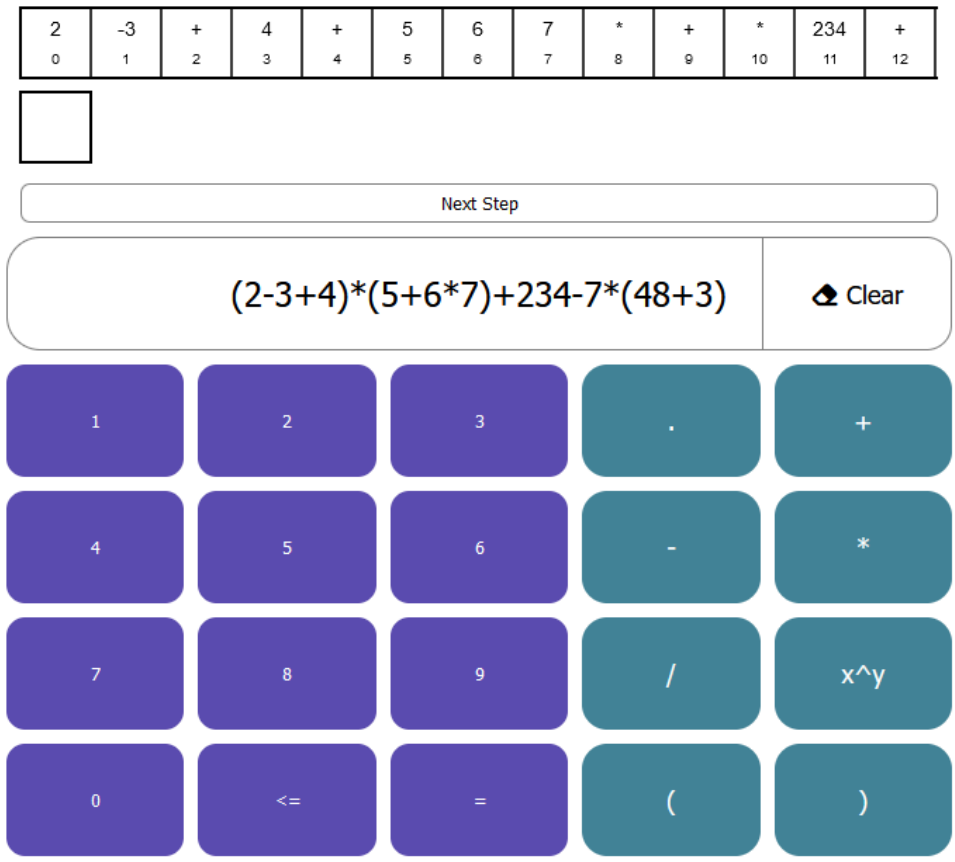
Client Javascript Applications

# Web application - Calculator

## Semester Project




## Introduction

This web application serves as an advanced calculator; It translates the mathematical expression into postfix notation, which allows it to evaluate complex sequences that include parentheses. A big part of the project is the animation mode, which allows the user to *peek* under the hood and observe the evaluation himself.

## Main functions

I implemented several different mechanisms, each major part is in its own JavaScript file. Each file is shortly explained in its own paragraph below.

### Calculator.js

The calculator itself - this file creates the buttons of the calculator and creates the correct event listeners, so the user can interact with the input field using those buttons. It also takes charge once the form has been submitted, by creating an instance of **ExpressionManipulator** and passing it to the newly created instance for evaluation, after asserting that only valid characters are inputted. After evaluation, it either sets the input field to the correct answer, or alerts the user that the expression is incorrect (e.g. 18+ or 7*(+-+1); 10/0 is not invalid, the result is just infinity).

### ExpressionManipulator

The first task of the **ExpressionManipulator** is to split the expression into individual nodes; `12+4` becomes [12,+,4], etc. Interesting edge cases are 10-2: before splitting, they are transformed into the arithmetically equal form `10+(-2)`. The other transformation is of constructs `-(12+5)`, they become `-1*(12+5)`. After those transformations, the expression is split into nodes and returned.

Next function used is ***translateToPostfix()***, which takes the array of nodes and returns an array where the expression is resorted into postfix notation. (Postfix)

Lastly, the expression is evaluated (Evaluation of postfix) using ***evaluate()***. This function is quite complex, because of the animation elements. If the animations are off, the function returns an immediately resolved promise with the correct result without any delays. However, if the animation mode is on, then the Promise is resolved upon finishing the animation. The animation is made possible by the **StackAnimator**.

### StackAnimator

The constructor of this class accepts an element, which is used as a parent for the stack. The skeleton of the stack is created using ***createSVG()*** which also appends it to the parent mentioned above. Once the skeleton has been created, ***constructStack()*** takes the array (also received in the constructor), and populates the stack with the values within it. It also exposes 3 other functions - ***popFirst()***, ***popLast()*** and ***append()***. All of them return promises that are resolved once the animation has been completed - this is necessary to ensure the correct sequential order of events. The code relies heavily on *transitionend* event, which is not fired in certain circumstances and edge cases, and so in key places a 'fake' is dispatched, to simulate it. **I don't have use for media controls, but they are listed in the requirements. To fulfill it, I also created a manual mode, which allows the user to walk the evaluation himself. I hope this counts.** I also use the Audio API for pop and push sounds.

### SettingsDialog

The settings dialog allows the user to change the theme of the application. Apart from that, the user can select his desired animation speed, or turn manual mode on instead. Last but not least, he can disable the audio or the animation completely. All of these settings are persisted thanks to the **localStorageAPI**. The different mechanisms outlined below read the local storage in order to perform their operations:

1. The **StackAnimator** checks if audio is allowed;
2. The **ExpressionManipulator** checks the speed preference and if the manual mode isn't engaged
3. The **SettingsDialog** controls the appearance - based on the selected option it swaps the correct theme.

The HTML was validated; some aria-label warnings were ignored, in sake of accessibility. The application does not need internet connection.

Here is a sample expression to try:

((2-3+4)*(5+6*7)+234-7*(48+3))^2