JavaScript

Calculator: Back end Part 3 a

By now, you should be comfortable with
1. Defining constants and using them
2. The or operator
3. If else statements
4. Manipulating a section of a html file using Javascript
5. Arrow functions
6. Event listeners.

That is something huge! Now, we want to expand our knowledge with what we already have. Logically, if you hit an operator key, the operator should be highlighted so that you know that the operator is active. To do so, we add the is-depressed class to the operator key, right below the operator keys if statement.

```
key.classList.add('is-depressed')
```

Now, we want to ensure that when you press 3, then 5, 3 is replaced by 5. To do this, we need to release the operators' depressed key. This can be done using a for loop. So, right below the is-depressed class that you had earlier on added, add this for loop.

```
Array.from(key.parentNode.children)
  .forEach(k => k.classList.remove('is-depressed'))
```

Do you understand what that code does? Discuss it and if you encounter any trouble call the tutor.

Next, we want to update the display to the clicked number/ button. Before we do this, we need a way to tell if the previous pressed button is an operator key. We can use a custom attribute, we'll call it *data-previous-key-type*. Now, chain calculator to dataset and the latter to the custom attribute we've made previousKeyType. A point to note, the previousKeyType is an operator, we'll want to replace the displayed number with clicked number. To do this, define a constant called previous key type and assign it the chained custom attribute we made before.

On the if statement that took care of displaying the key operators, we will add something extra on its parameter. If the dispayed number is zero or the the previous key type(the constant we just defined) is an operator, which is a string.

Now, the last part of the happy path section(I promise):

Normally, a calculate should calculate results of 2 values and an operator, I.e (2 + 2) and results of the calculation retuned. To know the second value that has been pressed, we'll define a constant called second value and equate it to the constant displayed number, we had earlier on defined it. This new constant replaces the message we had under the calculate data action. To get the first number, we need to store the calculator's displayed value before we erase it when we input the second value. One way to save this first number is to add it to a custom attribute when the operator button gets clicked. So, to get the operator we can use the same technique.

Right below the if statement that caters for the operator keys, chain calculator to dataset to firstvalue and assign it to the constant displayed number. Repeat the same process again, but instead of chaining it to first value, chain to operator and assign to the constant action.

Once we have the three values we need, we can now perform a calculation. Our calculate action should now have three constants, first is first value which is assigned to a chained method (calculator.dataset.firstvalue), second is operator assigned to calculator.dataset.operator and the last one is second value which is assigned to displayed number constant. In place of the console message, we will have the method display.textContent assigned to calculate which will have 3 parameters, which are the constants we just defined!

The final part of part 1, we need to create a calculate function which takes in three parameters: the first number(n1), the operator(operator), and the second number(n2) – this is an arrow function that will be a constant. So, If the operator is add, we will add values together. If the operator is subtract, we will subtract the values, and so on. I'll do the first one then you can continue! Ever heard of elseif? It would do you good here!

```
const calculate => {
let result = ''

 if (operator === 'add') {
 result = n1 + n2
}
  *do for the other operators*
  return result
}
```

Remember that firstValue and secondValue are strings at this point, and if you add strings together, you'll concatenate them I.e (1 + 1 = 11). So, before calculating the result, we want to convert strings to numbers. We can do so with the two functions parseInt and parseFloat.

parseInt converts a string into an integer.
parseFloat converts a string into a float (this means a number with decimal places).

For a calculator, we need a float. Now, edit your above codes to factor in parseInt and parseFloat. I'll do one, then you complete the rest.

```
if (operator === 'subtract') {
  result = parseFloat(n1) - parseFloat(n2)
  }
```

So far, so good! We are done with our happy path! Now, take a break, we are moving on to bigger things now!