## **Calculator**

### **Introduction**

You made it! By now you should have a *really* firm grasp on the fundamentals of JavaScript. Of course there’s plenty more to learn, but you should be able to create quite a bit at this point. Our final project is going to combine everything you’ve learned so far: you’re going to make an on-screen calculator using JavaScript, HTML, and CSS.

As usual with these things, there are elements of this project that are not going to be trivially easy for you, but if you’ve been following the course so far, you definitely have everything you need to finish it. We’re going to walk you through the various steps you can take, but again, how you actually implement them is up to you!

Important Note: Before you get started with this calculator project, we need to cover a word of warning. As you look into how to evaluate complex mathematical statements in JavaScript, you will likely come across the tantalizing [eval()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval) function. However, this function can be very dangerous and [should not ever be used](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/eval#Never_use_eval!)! You’ll need to build your own functions to evaluate expressions as part of this calculator project. On the same note, when researching how to calculate expressions for this project, you may encounter solutions that suggest that you return a new Function() that evaluates a string. Similarly to eval(), this should not be used [due to potential pitfalls of evaluating insecure data](https://stackoverflow.com/questions/4599857/are-eval-and-new-function-the-same-thing). Besides, where’s the fun in solutions that do all the work for you? Let’s get to it!

### **Assignment**

Here are some use cases (abilities your project needs to have):

1. Your calculator is going to contain functions for all of the basic math operators you typically find on simple calculators, so start by creating functions for the following items and testing them in your browser’s console.
   1. add
   2. subtract
   3. multiply
   4. divide
2. Create a new function operate that takes an operator and 2 numbers and then calls one of the above functions on the numbers.
3. Create a basic HTML calculator with buttons for each digit, each of the above functions and an “Equals” key.
   1. Do not worry about wiring up the JS just yet.
   2. There should also be a display for the calculator, go ahead and fill it with some dummy numbers so you can get it looking right.
   3. Add a “clear” button.
4. Create the functions that populate the display when you click the number buttons… you should be storing the ‘display value’ in a variable somewhere for use in the next step.
5. Make the calculator work! You’ll need to store the first number that is input into the calculator when a user presses an operator, and also save which operation has been chosen and then operate() on them when the user presses the “=” key.
   1. You should already have the code that can populate the display, so once operate() has been called, update the display with the ‘solution’ to the operation.
   2. This is the hardest part of the project. You need to figure out how to store all the values and call the operate function with them. Don’t feel bad if it takes you a while to figure out the logic.
6. Gotchas: watch out for and fix these bugs if they show up in your code:
   1. Users should be able to string together several operations and get the right answer, with each pair of numbers being evaluated at a time. For example, 12 + 7 - 5 \* 3 = should yield 42. An example of the behavior we’re looking for would be [this student solution](https://mrbuddh4.github.io/calculator/). Your calculator should not evaluate more than a single pair of numbers at a time. If you enter a number then an operator and another number that calculation should be displayed if your next input is an operator. The result of the calculation should be used as the first number in your new calculation.
   2. You should round answers with long decimals so that they don’t overflow the screen.
   3. Pressing = before entering all of the numbers or an operator could cause problems!
   4. Pressing “clear” should wipe out any existing data.. make sure the user is really starting fresh after pressing “clear”
   5. Display a snarky error message if the user tries to divide by 0… don’t let it crash your calculator!
7. EXTRA CREDIT: Users can get floating point numbers if they do the math required to get one, but they can’t type them in yet. Add a . button and let users input decimals! Make sure you don’t let them type more than one though: 12.3.56.5. It is hard to do math on these numbers. (disable the decimal button if there’s already one in the display)
8. EXTRA CREDIT: Make it look nice! This can be a good portfolio project… but not if it’s UGLY. At least make the operations a different color from the keypad buttons.
9. EXTRA CREDIT: Add a “backspace” button, so the user can undo if they click the wrong number.
10. EXTRA CREDIT: Add keyboard support!

## Solutions

<https://michalosman.github.io/calculator/>

<https://rlmoser99.github.io/calculator/>

<https://captain-usopp.github.io/Calculator/>

<https://calculatorjs-1.anastazy666.repl.co/>

<https://artis-dev.github.io/calculator/>