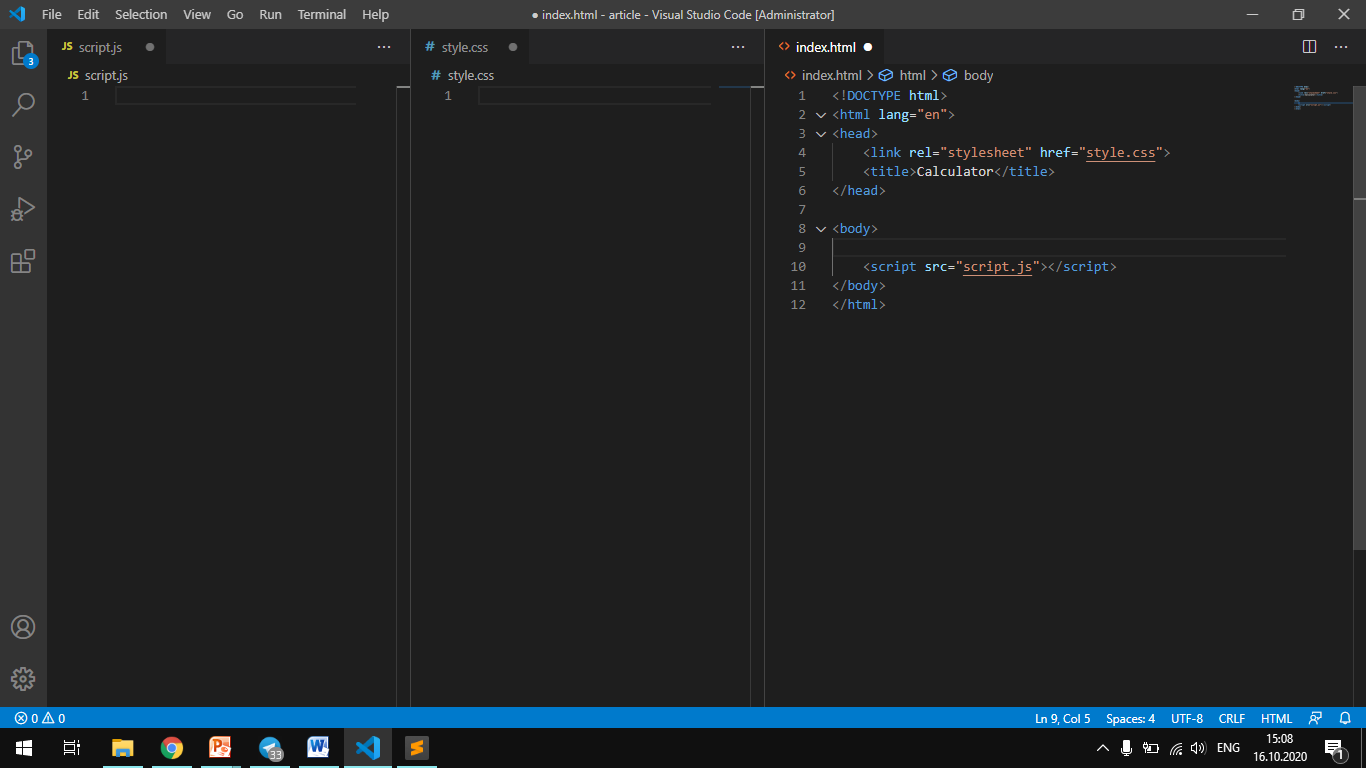
Student: Abibulla Gulzada ID: 190103229 Bachelor: Computer Science

**Article #2**

**How to create calculator with HTML, CSS, JAVASCRIPT?**

In this article, I will show you How to write a simple calculator in Java Script with priority support. Priority support in the calculator allows you to perform complex mathematical expressions with all the signs of operations, as well as use parentheses to increase the priority.

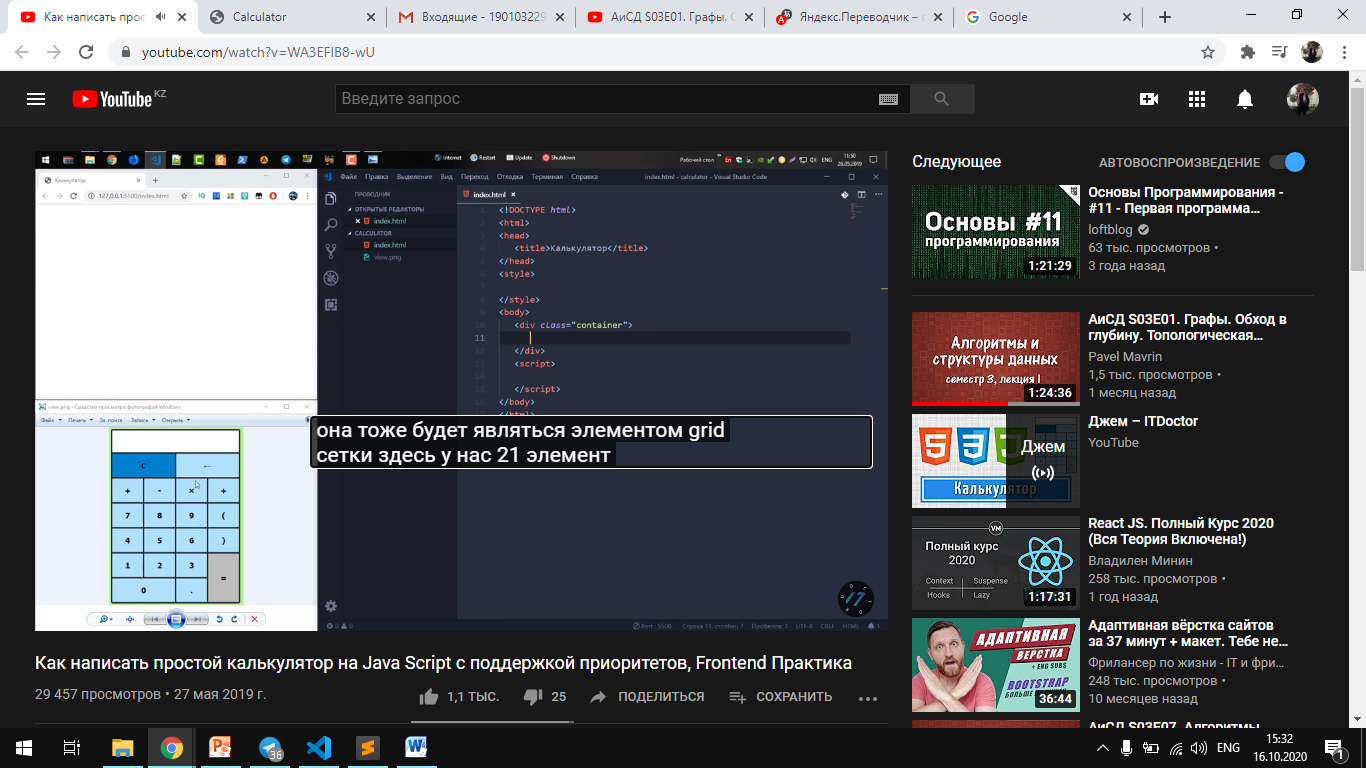
The first thing we will do is create index.html, style.css, script.js.



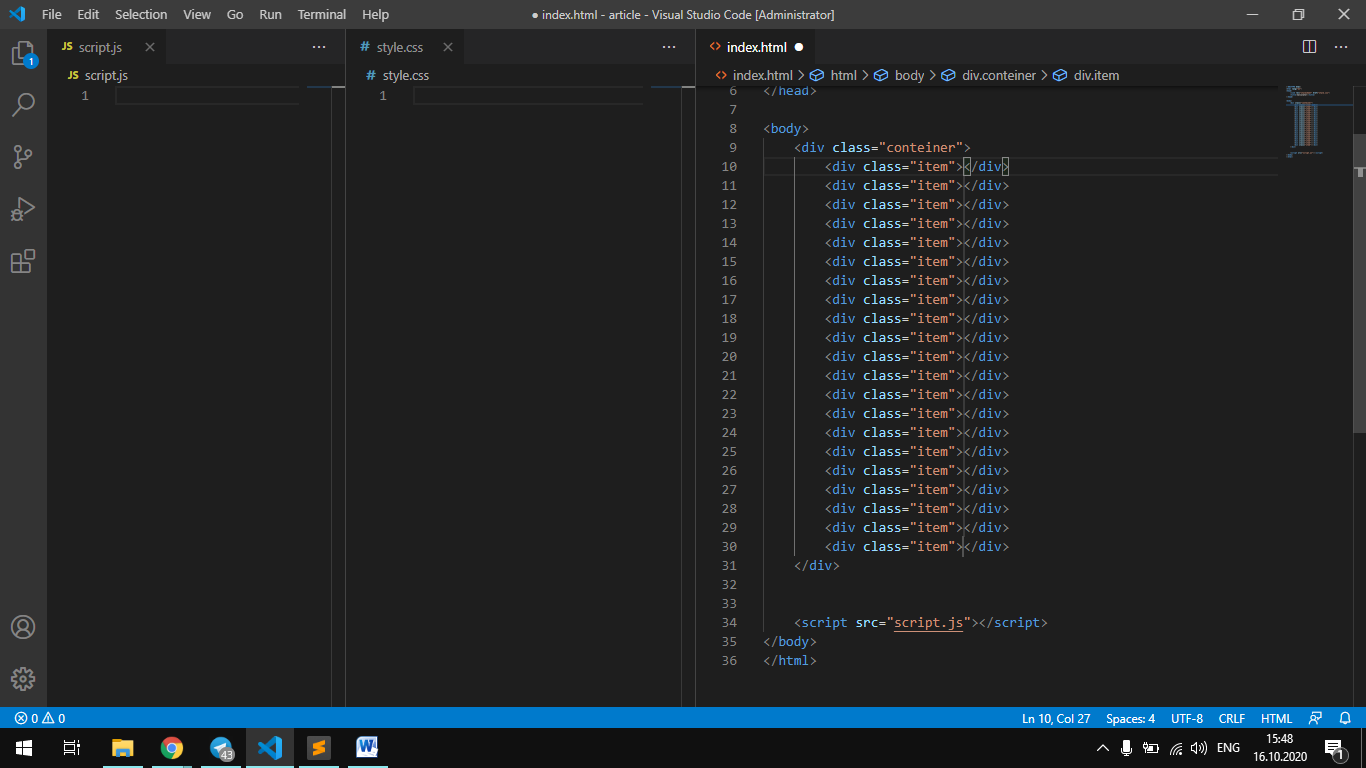
As you can already see all the links to style.css and script.js already attached.

First, we'll create html markup, and then add the styles we need.

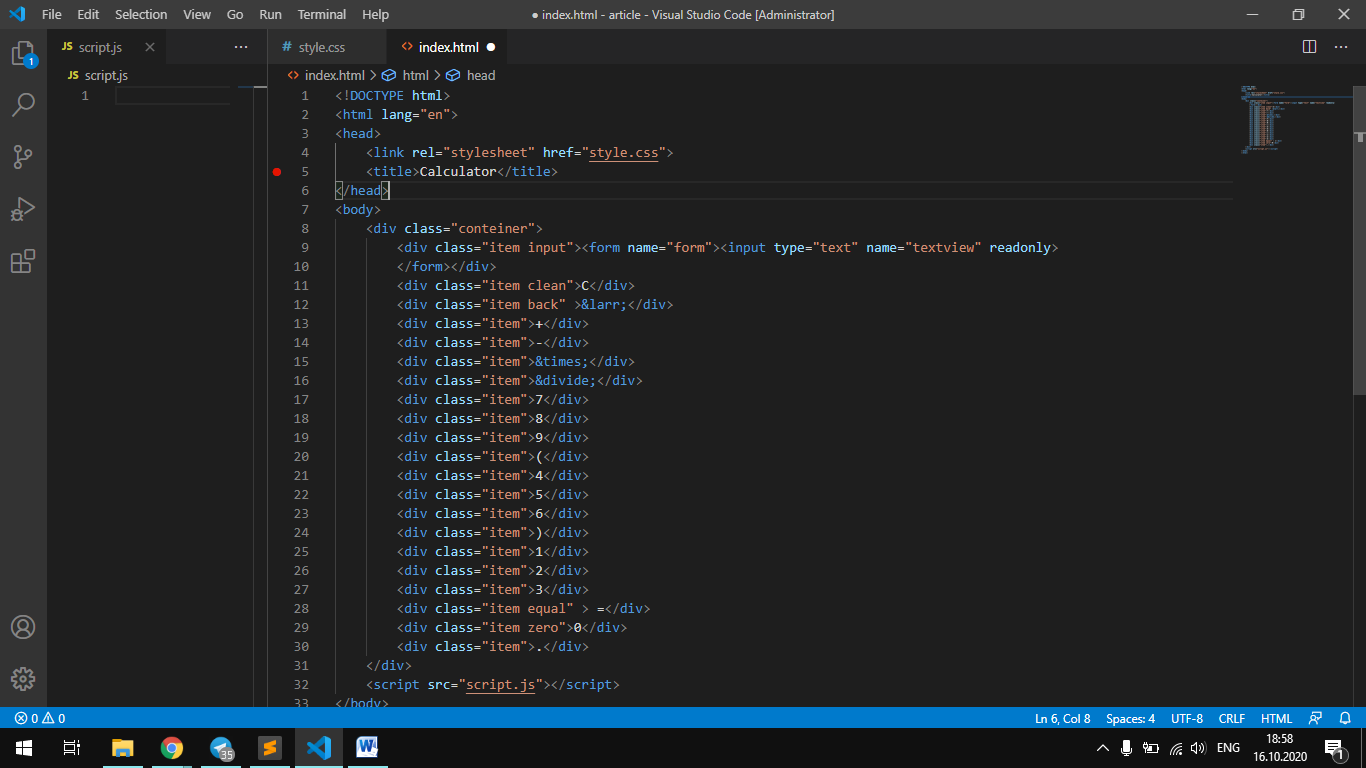
Next, we will create the functionality using our javascript. Let's create a div tag with the conteiner class. And inside it we will put all the elements of our buttons. Then we will transform the text field where all the results will be and then we will style it using CSS GRID.And just practice this element. Here is our layout of the calculator, according to it, we'll use a CSS GRID and JAVASCRIPT functions.



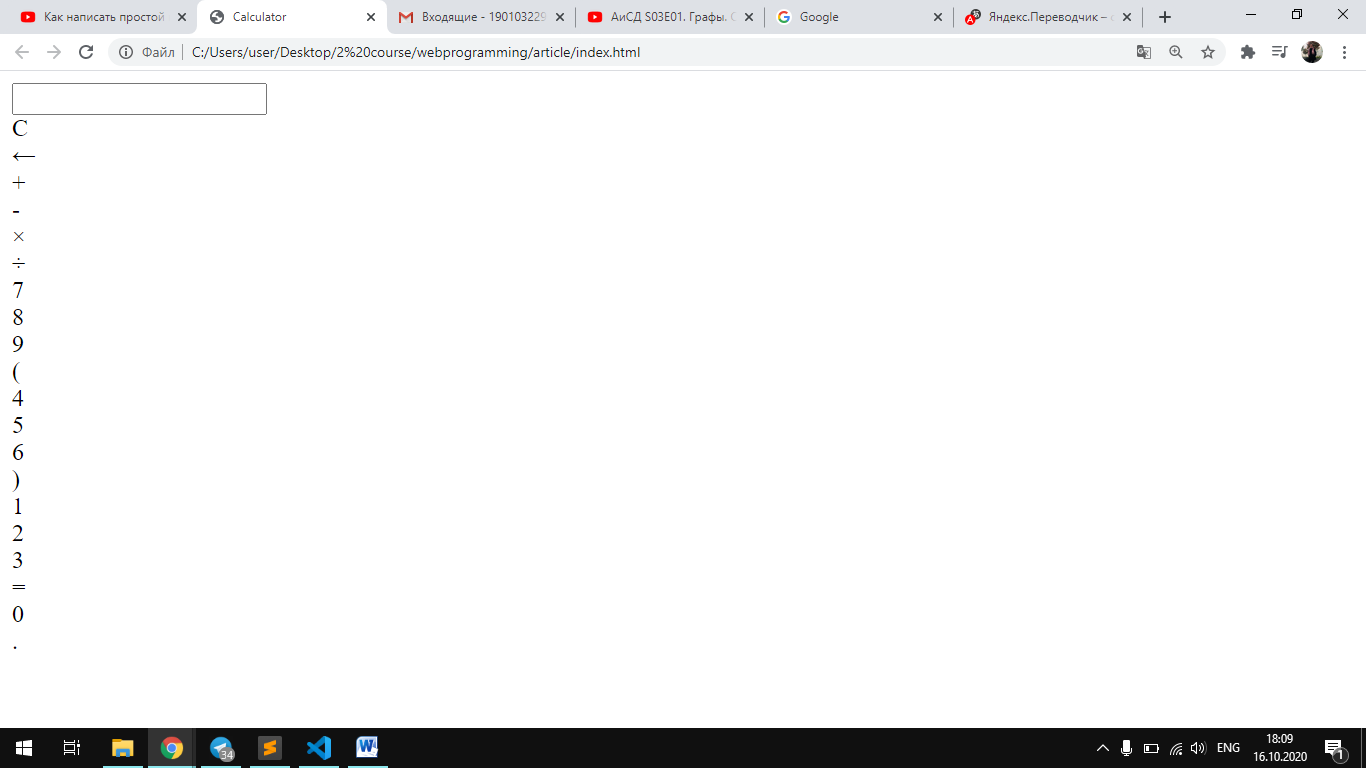
Looking at the layout, we need to add as many elements as we have buttons here. We also take into account our input field, since here we will display all the text. It will also be a grid element of the grid. We have 21 elements in the layout, so we create a div tag with the item class and multiply it, so we will make our work easier.



So far, nothing will be visible on our site, but now we need to add the corresponding text in each tag, namely all the operation signs, numbers, the sign exactly, and other symbols for the calculator. In the first cell, we need to add a form called form and inside it will be input type=text. Also, the name for this input is set to name="textview" and set readonly. Then we write numbers and symbols in each cell as in the layout. The texts &larr, &times, &divide are html notes. Since we don't find these characters in some texts, they are embedded in html.



So in the site we have buttons lined up in one column. Then we will format everything using CSS.



But first of all you need to add additional classes for buttons that we will write separate functions in script.js

In our layout, approximately all the buttons look single, we added them for buttons C, removing buttons and equal buttons, zero buttons, input buttons. We will style these 5 classes separately.

In style.css, we set the General style for the site, that is, let the background-color be green. Let's add the user-select: none property.this property is necessary so that you can't select a character on our page when you click or select it with the mouse. If these properties are not set, the symbols will be highlighted and will interfere with our calculator operation.

Let's continue our stylization: we will transform the grid.

grid-template-columns or grid-template-rows

Defines columns and rows in the grid using a space-separated list of values. The values represent the size of the track, and the spaces between them are the grid lines.

fr - fraction (flexible size). Can only be used for max. values.

grid-column-gap, grid-row-gap, or grid-gap

Defines the size of the line width between rows, between columns, or for two parameters at the same time.

Values: line-size - size value, for example in px

Examples: grid-column-gap: 10px; grid-row-gap: 20px; grid-gap: 15px; grid-gap: grid-row-gap grid-column-gap

grid-column , grid-row

Abbreviations for grid-column-start + grid-column-grid-row-start and + grid-row-end, respectively

Values: grid-column: start line / end line; grid-line: start line / end line; grid-column: starting line / the value of the range; grid-row: run rows / interval value ;

Examples: grid-column: 2 / span 3; grid-string: myLine1 / -1;

justify-content (1) align-content (2) place-content

Aligns the grid inside the container. (1) aligns elements horizontally. (2) aligns elements vertically.

Values: start - allocate all the elements at the beginning of the cell (left / top) end - puts all elements in the end cell (right / bottom) center - puts all elements in the center of the cell stretch - stretches all the elements to the full width / height container space-around - the same space between the elements, and Polarstern indents on the edges space-between - the same space between elements, without margins on the edges space-even - the same space between elements and full-size margins on the edges

Examples: align-content: center; justify-content: end; place-content: start; place-content: align-content justify-content;

Cursor property. Sets the shape of the cursor when it is within the element. The cursor type depends on the operating system and the settings you set. Before you take the opportunity to change the cursor's appearance, decide whether it will be used in the appropriate way. For many users, such changes can be misleading, when, for example, instead of the traditional hand that appears when you hover over a link, something else appears. In most cases, it's best to leave things as they are.

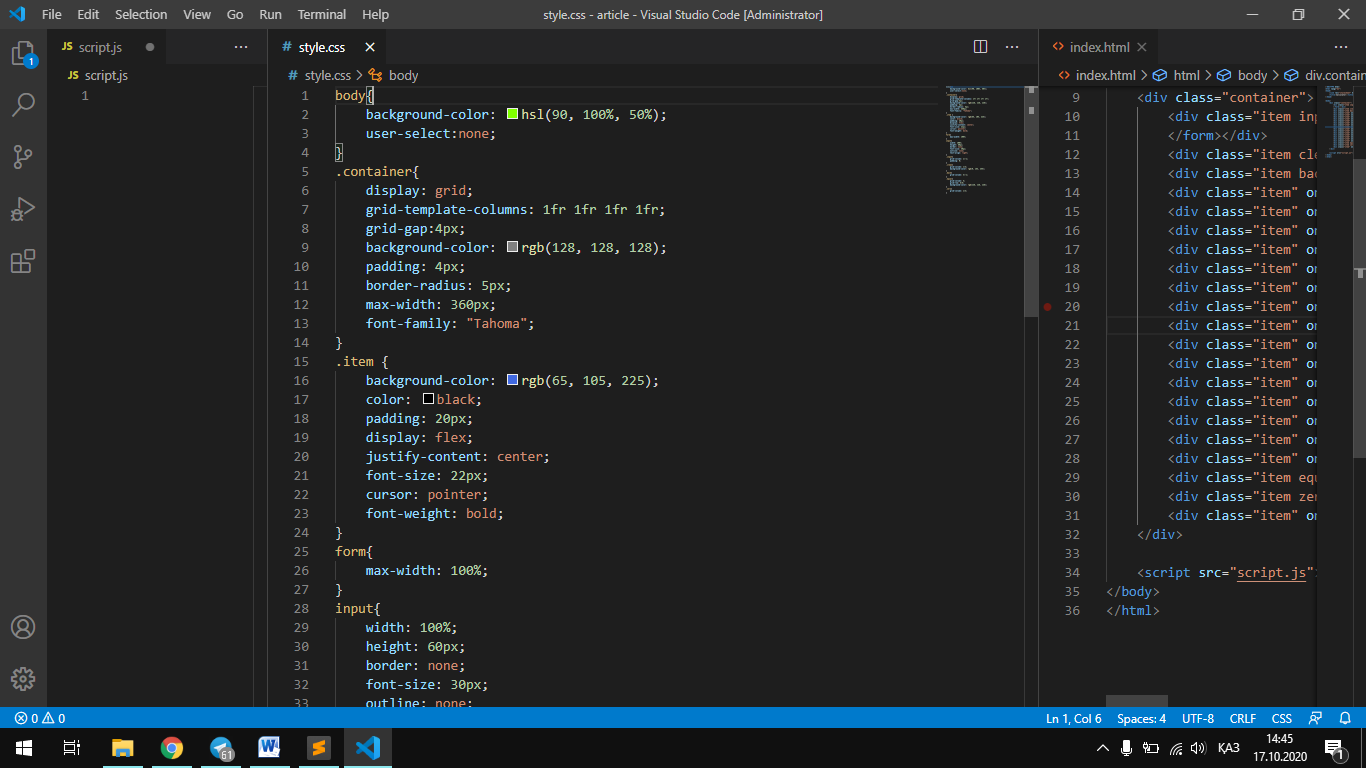
Syntax

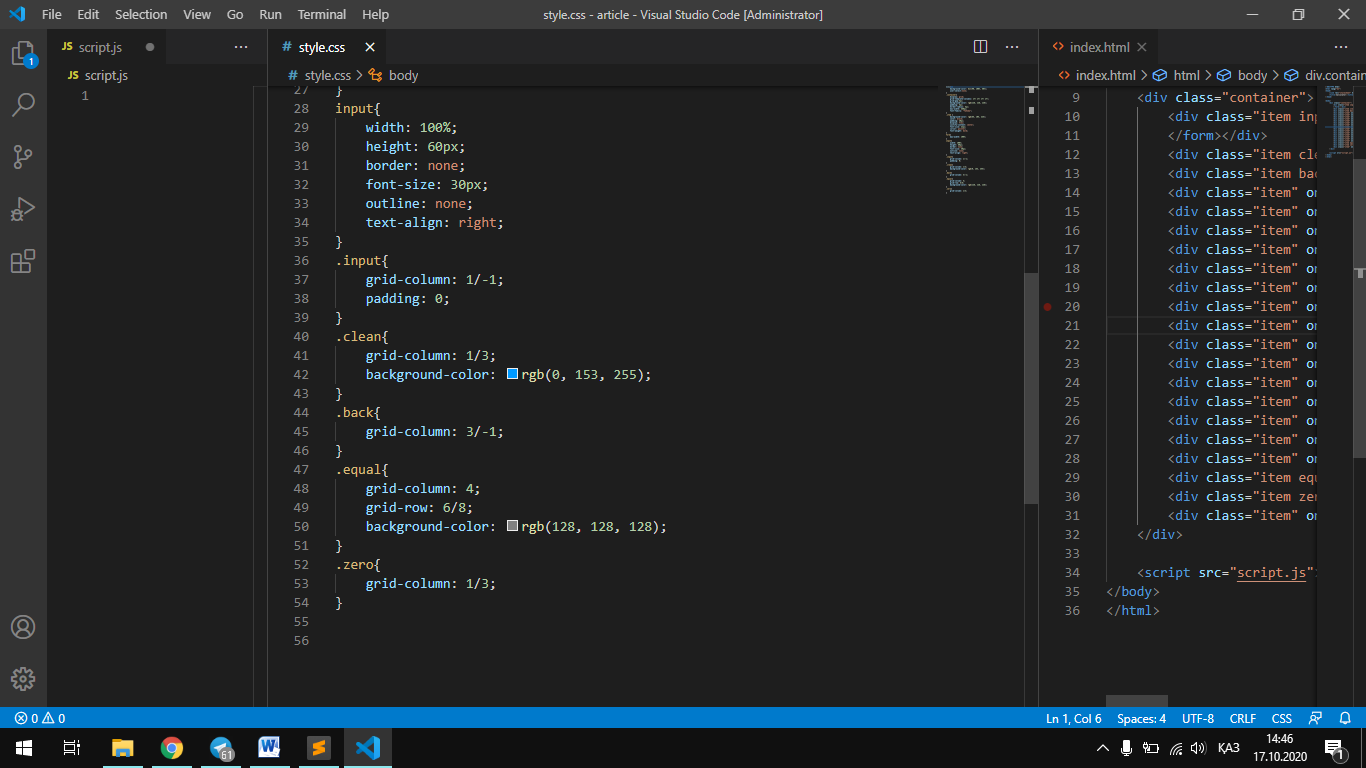
cursor: [url('path to the cursor'),] | [ auto | crosshair | default | e-resize | help | move | n-resize | ne-resize | nw-resize | pointer | progress | s-resize | se-resize | sw-resize | text | w-resize | wait | inherit ]

Values

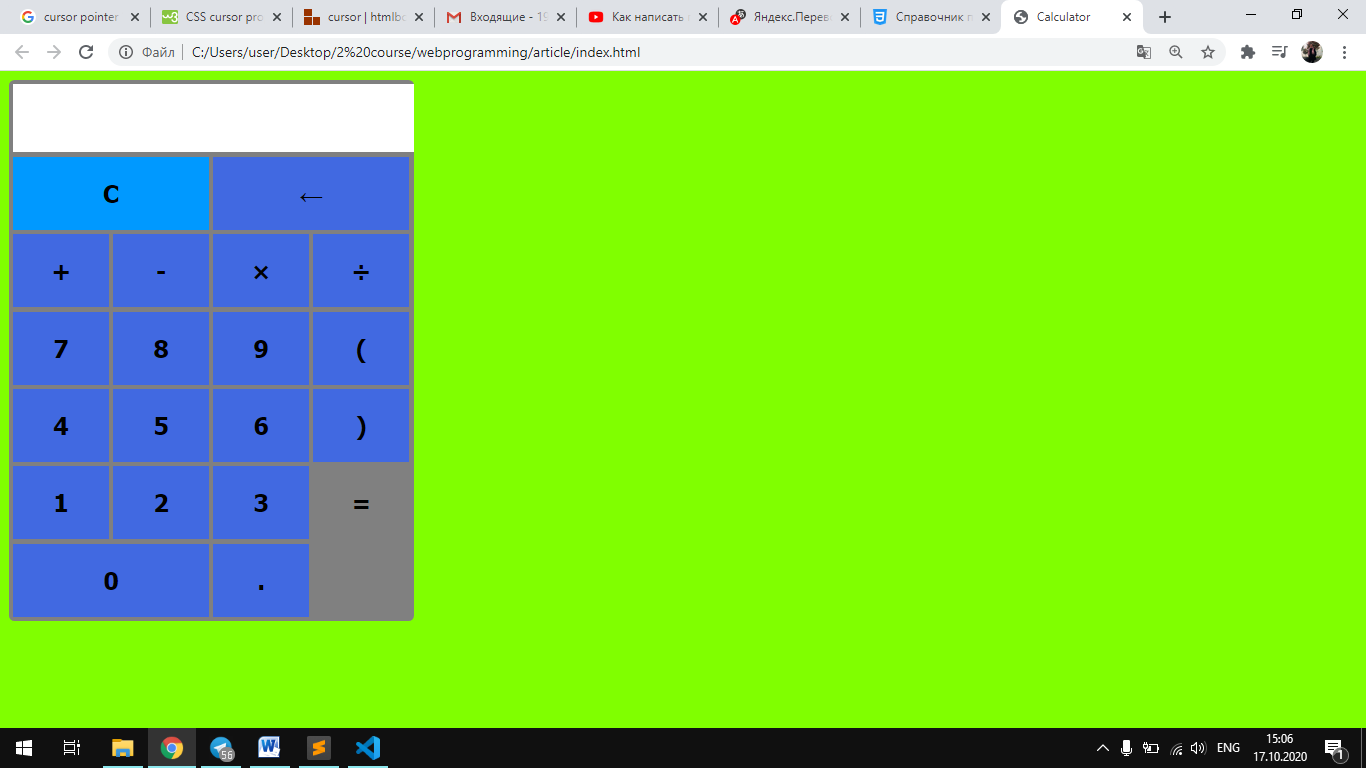
url - Allows you to set your own cursor.to do this, specify the path to the file with the cursor. auto - The appearance of the default cursor for the current element. inherit - Inherits the value of the parent.

Finally, we will add some properties for the decoration, so that our calculator looks no less beautiful. In total our css code will look like this:





The calculator will display on our website as follows:



Now let's move on to our script.js

At the moment, our website just shows a calculator.Now we will write functions to make our calculator work.

In addition, we will write functions. Namely these are functions: 1) delete the entire text; insert() 2) delete one character; back() 3) add one specific character; clean() 4) the equals function, which will calculate everything that we entered. equal()

The insert() function will take a single argument value and do the following:

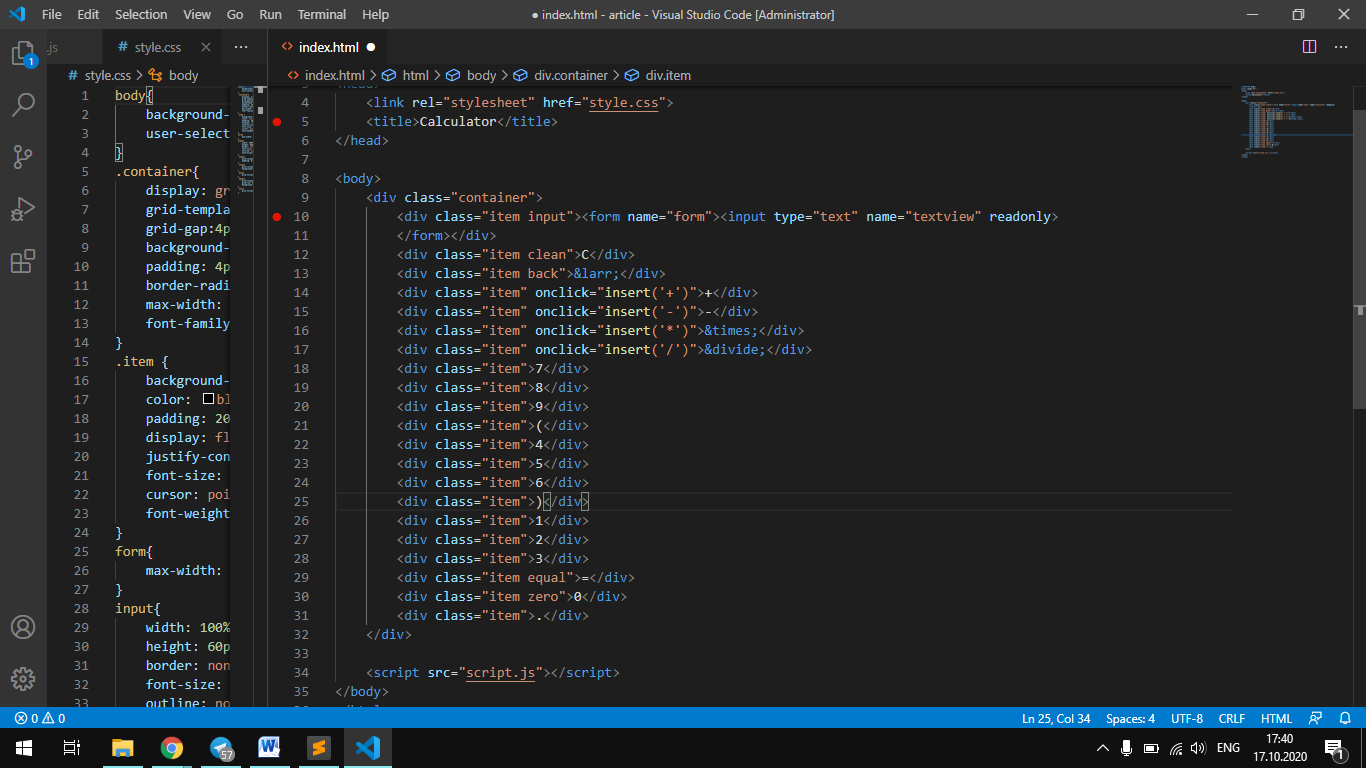
function insert(num){

    document.form.textview.value=document.form.textview.value+num;

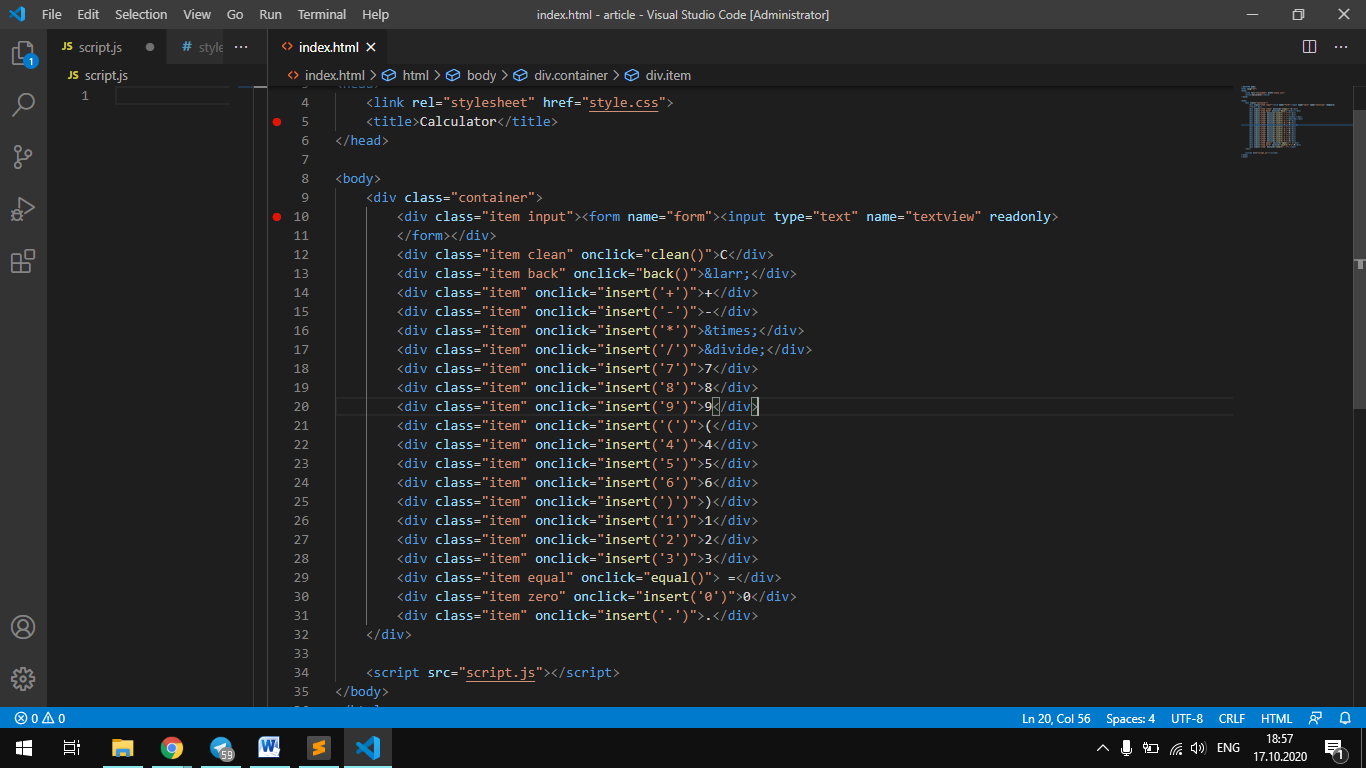
}

In the document we select our form in it we will have a textview and in the value of this textview we will set the same thing here but we will also add our *num*. In order to activate our function, we add some adjustments in the html code. For example, for the characters ‘+’ , ‘ –‘ , ‘ /’ , ‘\*’ .

In html, we add the onclick method for handling the left-click event to these characters, and here we call our insert('value') function, and insert these characters in the value place.



Now we do the same for all other symbols, and for clearing, deleting a symbol, and counting the result, we will make 3 separate functions. And so we insert these methods in the places we need.



Now let's move on to the other three functions.

The clean() function is the easiest, since we equate textview with an empty string.

function clean(){

    document.form.textview.value="";

}

For our back() function, we create a new variable that will contain all the data, namely ‘document.form.textview.value’.

We must take our variable from it to get a substring using the ***substring ()*** method.

function back(){

    var exp=document.form.textview.value;

    document.form.textview.value = exp.substring(0, exp.length-1);

}

Let's move on to the last function, which is counting the result. Here we will also have a variable that contains the current state of the text string and we check the condition: if the variable is not empty, we will set the result in this field through which we will apply the eval(exp) method.

function equal(){

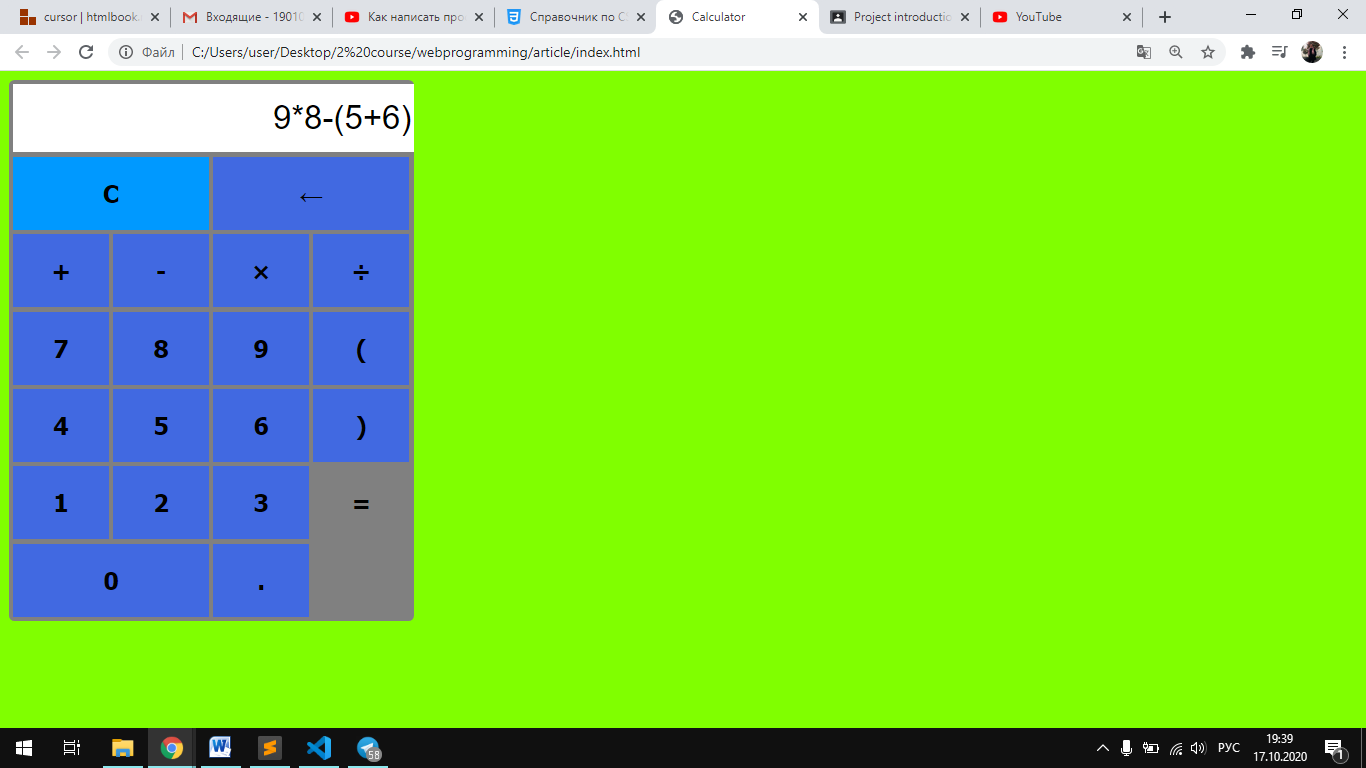
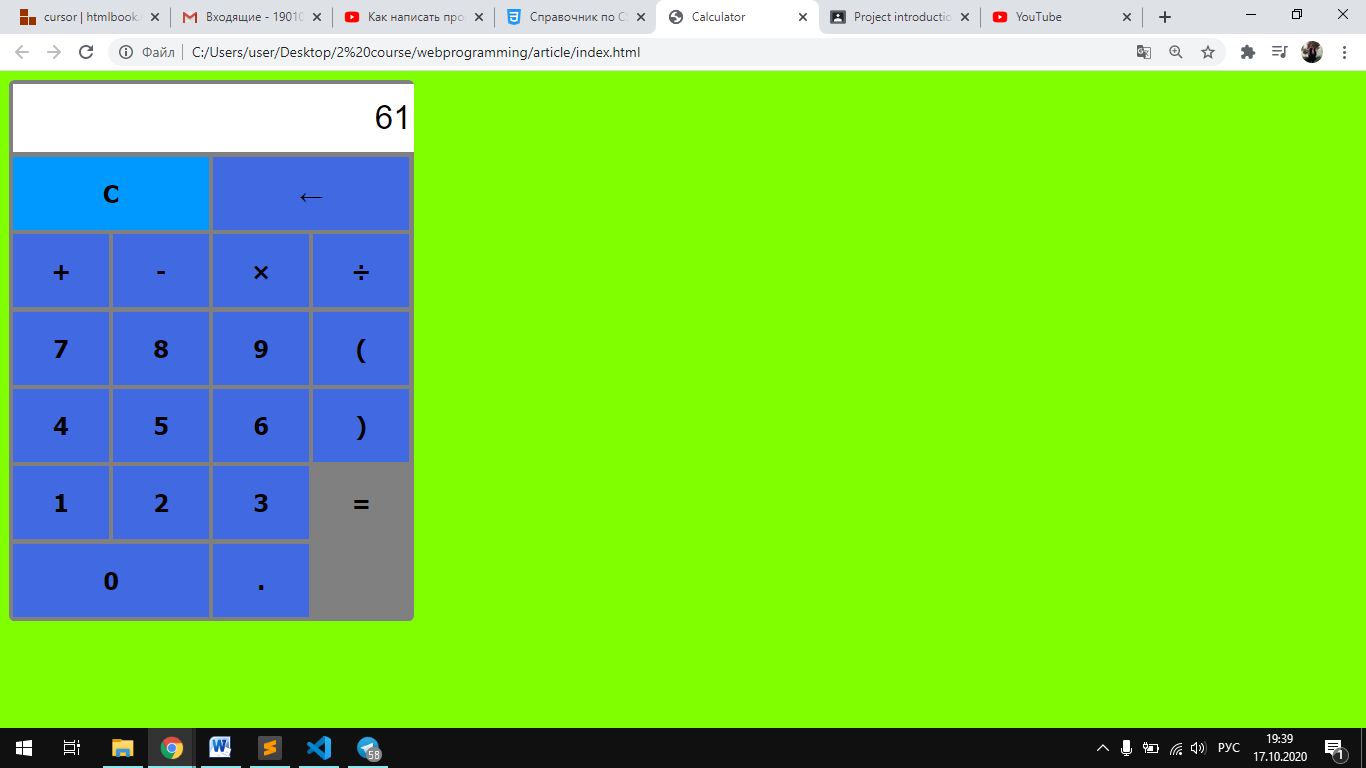
    var exp=document.form.textview.value;

    if(exp){

        document.form.textview.value=eval(exp);

    }

}

The eval() method takes this string in the exp variable and calculates the result. Everything is clear and simple. eval() methods support all characters, parentheses, fractional characters, and so on. In our web page, the calculator now works. 

**Conclusion**

In this article, you can learn how css grid works, also by writing very little code in html, we can manage everything in css. And most importantly, you have mastered some ready-made methods in javascript. In such a simple program as a calculator, we can improve our code. as you can see, we haven't written much code.

Source of information

1. <https://www.youtube.com/watch?v=WA3EFIB8-wU&t=867s> the video we used for mastering the material;
2. <https://morphismail.github.io/css-grid-manual/> css properties that we applied;
3. <http://htmlbook.ru/css/cursor> a little material for cursor.