**Práctico 3:**

Implementación de una calculadora en



Alumno: Carolina Garaycoechea

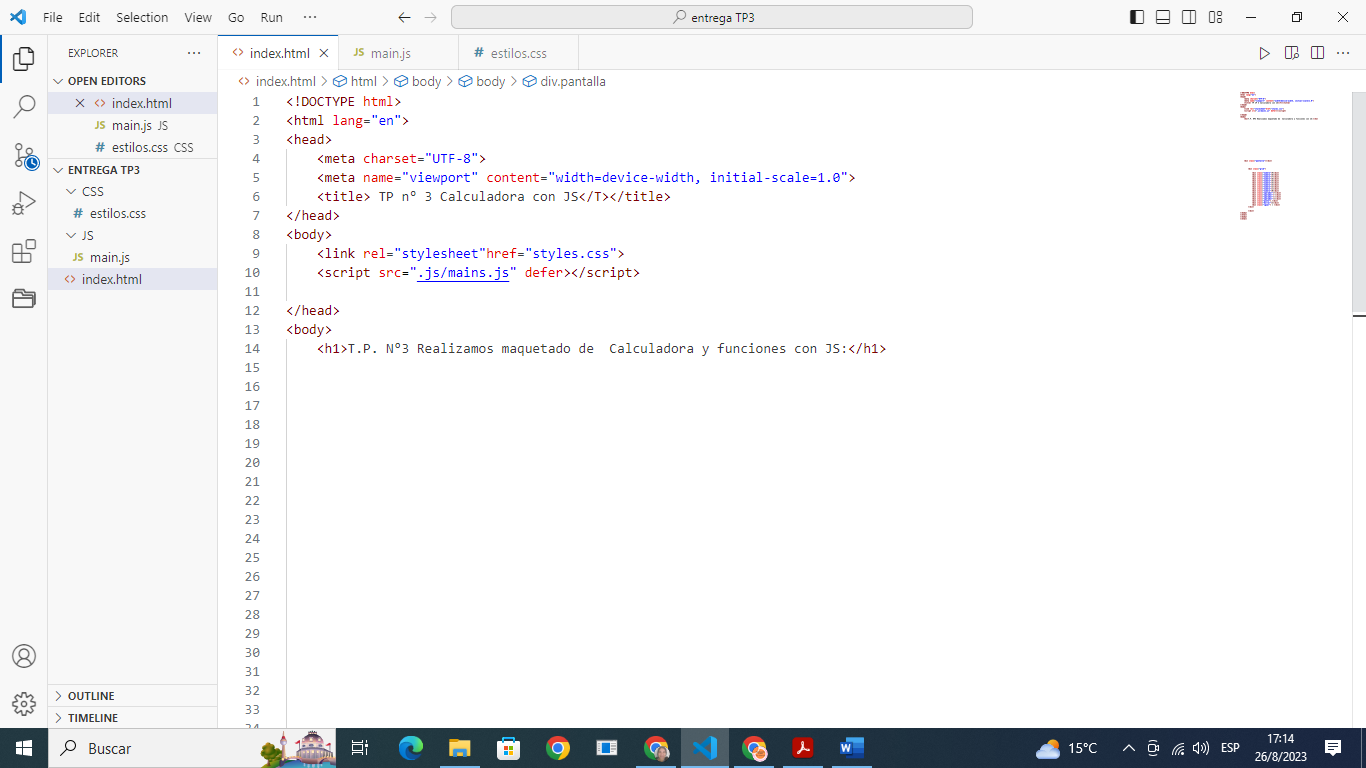
Materia: Practica 1

Profesor: Luis Romano

url Github:

Maquetado de la Calculadora

Comenzamos…



Creamos el archivo index.html, y los archivos de Css y Js en su respectiva carpeta para una mejor organización. - Como primera instancia con los atajos del VSC armamos la estructura del documento Html.

Introducimos también en el html el enlace a css con

<link rel="stylesheet" href="CSS/estilos.css">

Y también con el script para JS esto nos permite acceder a los enlaces desde el index cuando llamemos al archivo.

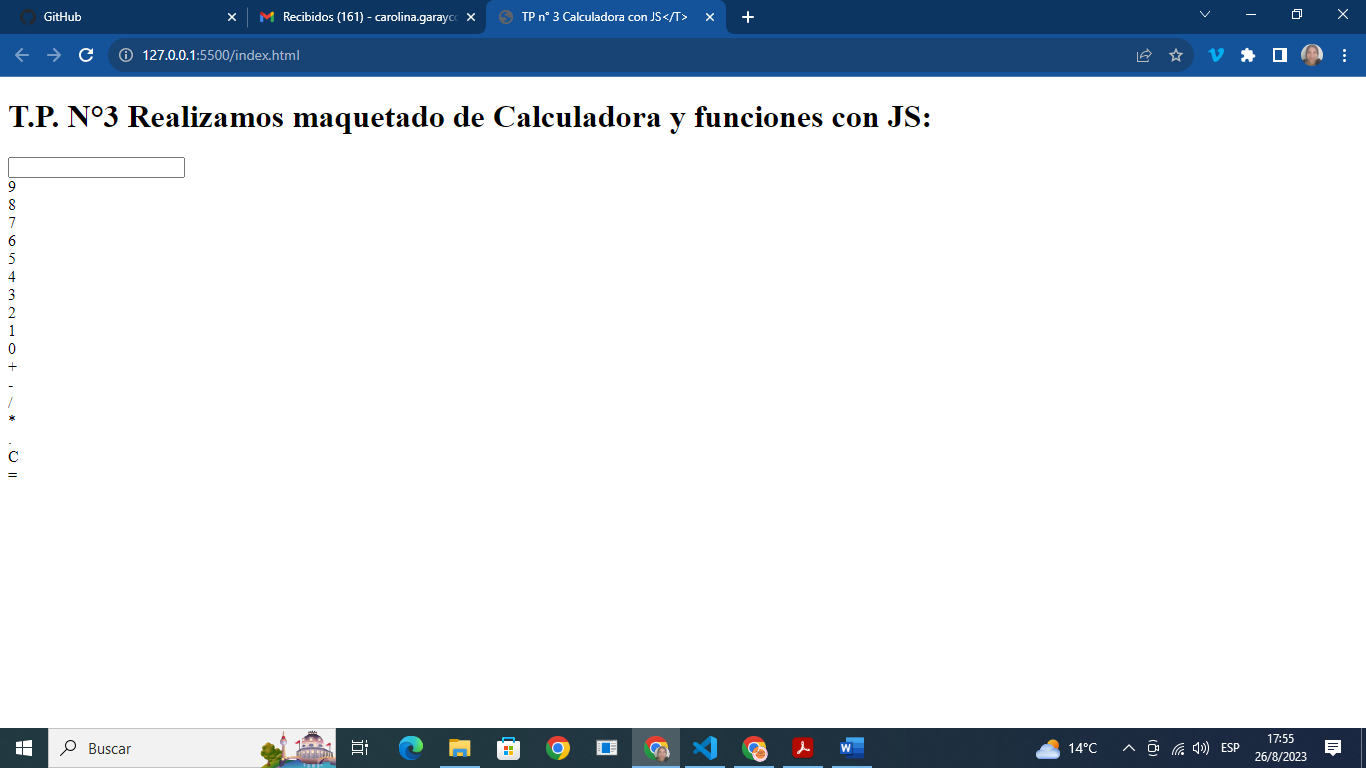
<script src="JS/main.js" defer></script>

Con las etiquetas div vamos dando formato a las cajitas en este caso pantalla, con la palabra clave class vamos a llamar a las clases en el archivo de estilos de css.

<div class="pantalla">

        <div class="resultado"></div>

Una vez realizado el esqueleto en html con las etiquetas div procedemos a darle forma y color con css



 </div>

    <div class="numero">7</div>

    <div class="numero">8</div>

    <div class="numero">9</div>

    <div class="operador">\*</div>

    <div class="numero">3</div>

    <div class="numero">4</div>

    <div class="numero">5</div>

estilo al html con la carpeta ya enlazada, como hemos aprendido existen tres tipos de selectores el primero es con la etiqueta body por ejemplo

body .h1 {

    font-family: 'Rokkitt', serif;

    background-color: rgb(223, 227, 230);

El segundo es con el punto que se refiere a clase como se muestra en la siguiente seccion de código, con display grid acomodamos las cajitas y con grid-template-columna organizamos la separación en columnas y con grid-template-rows oragnizamos las filas.-

.pantalla {

    display: grid;

    grid-template-columns:100px 100px 100px ;

    grid-template-rows: 100px 100px 100px;

    gap: 5px;

    width: 80px;

}

Y el tercero que es con un id y llamaremos desde css con numeral por ejemplo

{

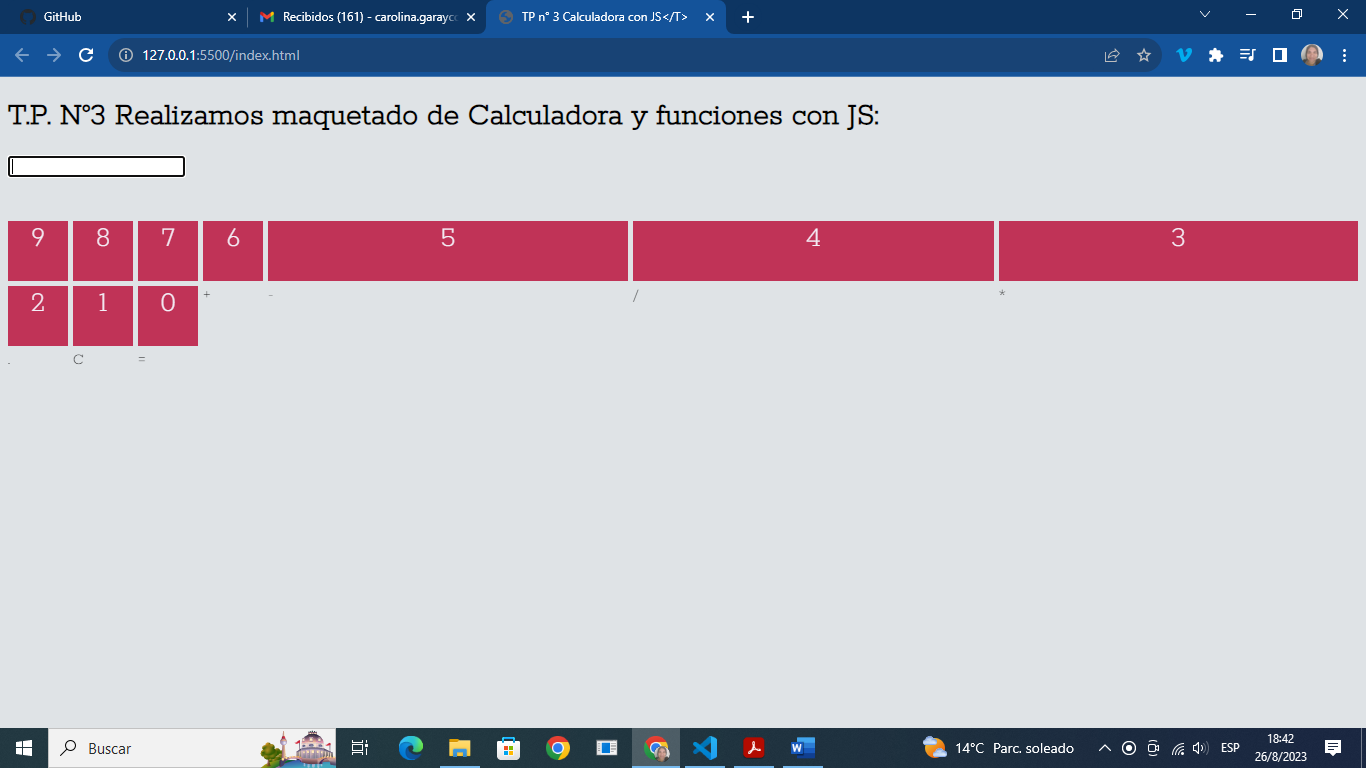
#contiene

}

También utilizamos Google fonts dándole formato al texto y/o números dado el caso, y ubicamos las cajas donde se ubican los números.

font-family: 'Rokkitt', serif;

todavía falta seguir acomodando posiciones, pero ya podemos ver que los números cambiaron su formato con Google fonts



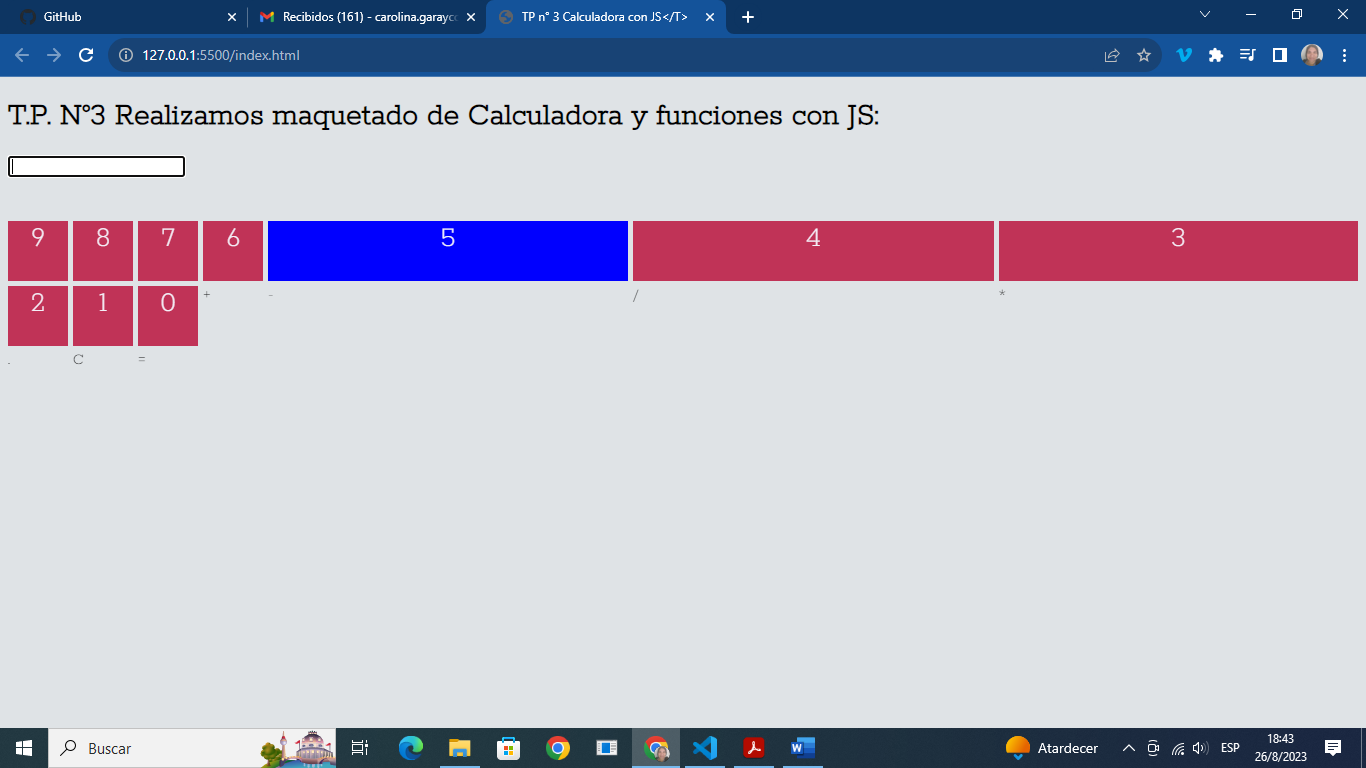
Cuando pasamos el puntero sobre las cajitas cambia de color, gracias a hover

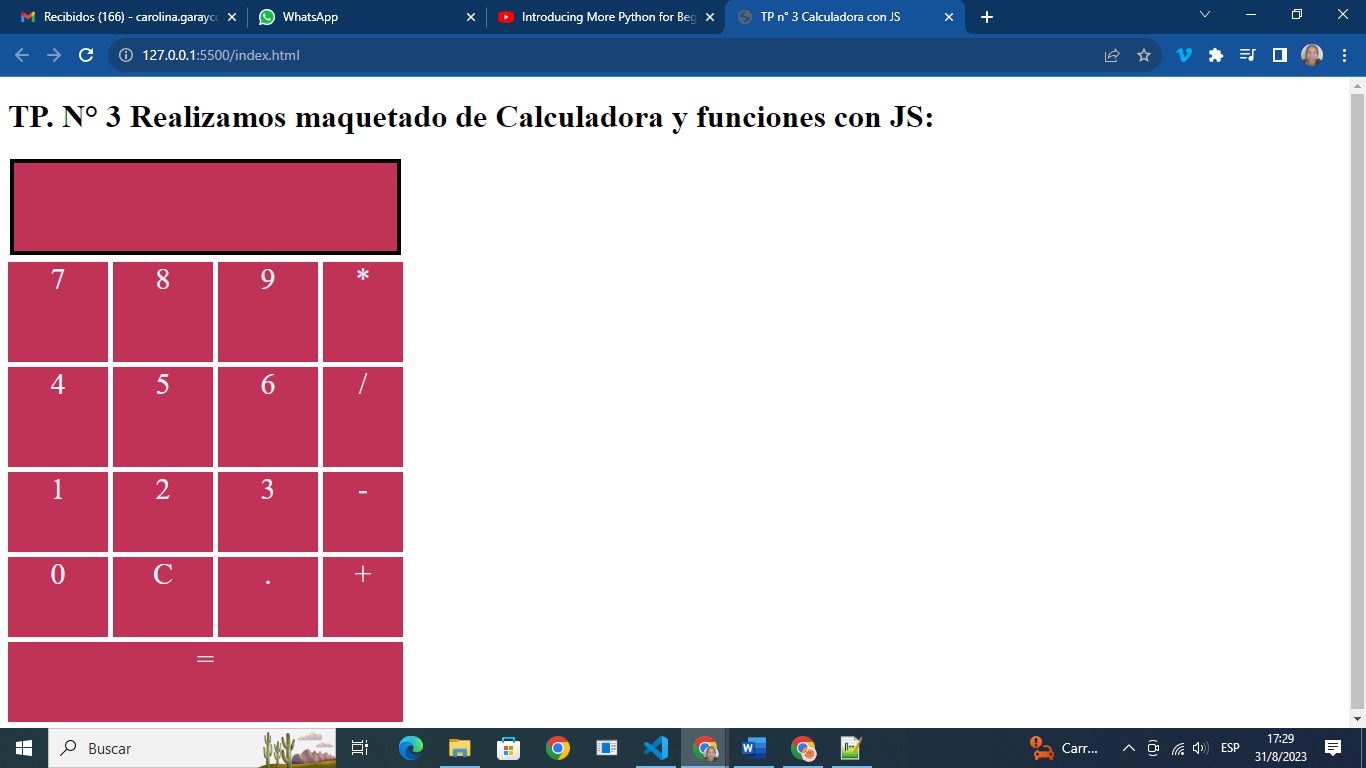
 .pantalla .numero:hover{

    background-color: blue;

    cursor:pointer;

 }

acomodamos los botones con css utilizamos la clase según el nombre designado .operadores, .igual, .cero, .borrar.-



Javascript:

Ahora aplicaremos la lógica para hacer funcionar la calculadora, ya hemos enlazado el script de js, empezaremos declarando una constante;

const numeros = document.querySelectorAll(".numero");

/\*console.log(numeros); // para mostrar en pantalla si funciona \*/

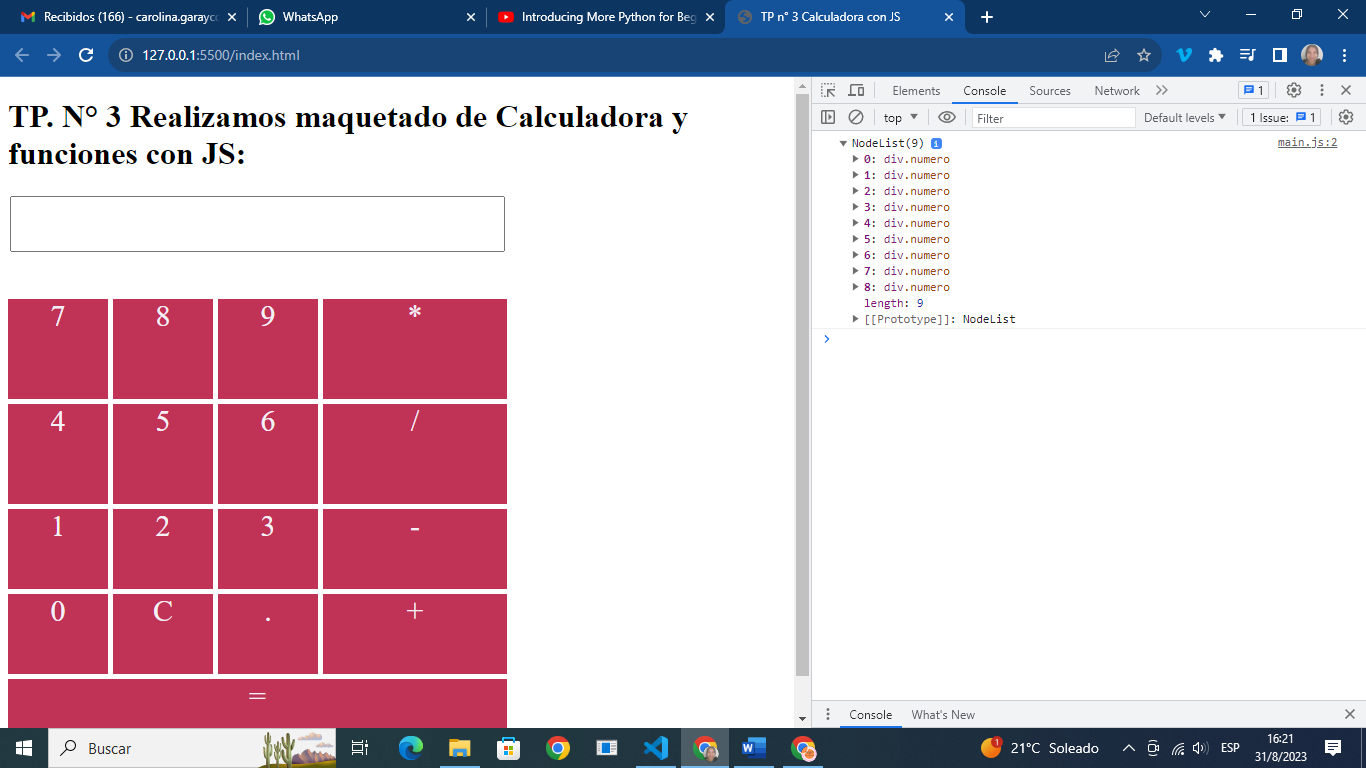
const operadores = document.querySelectorAll(".operador");

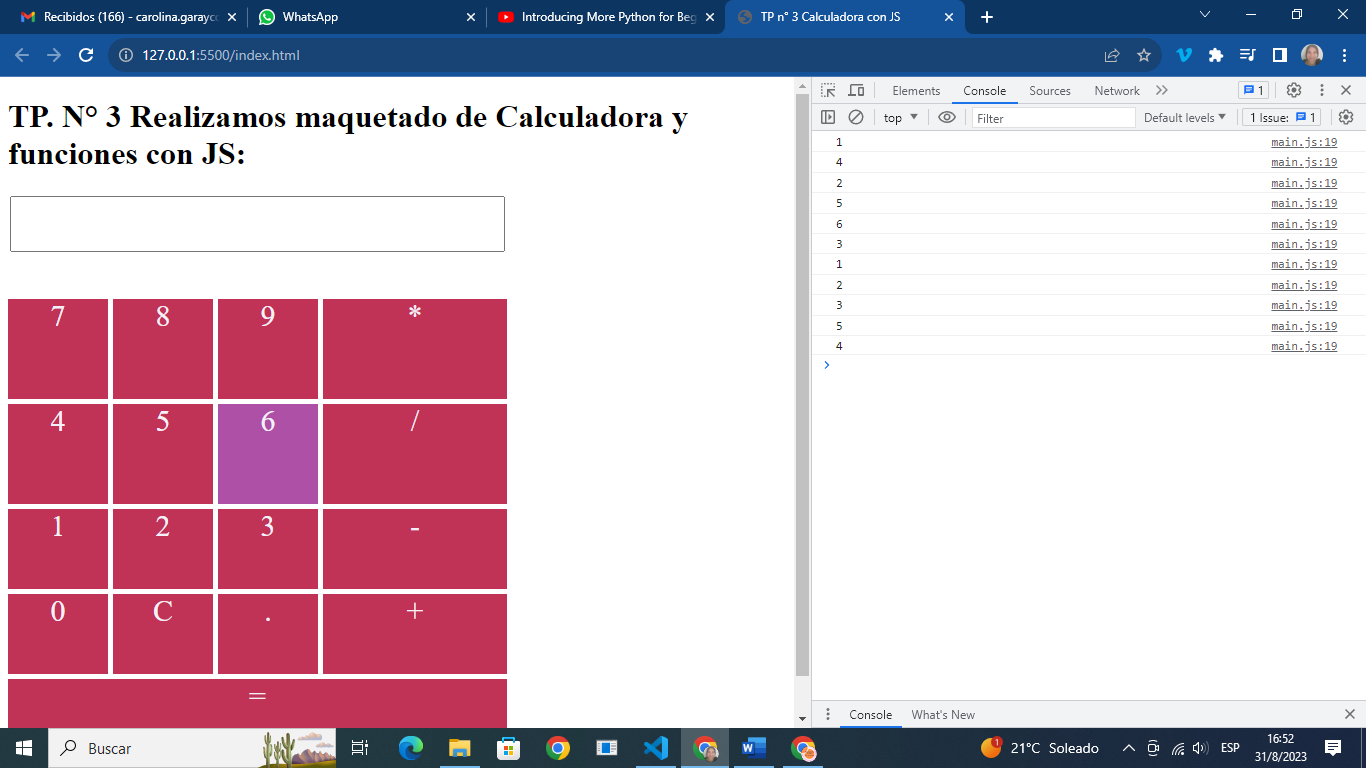
const pantalla= document.querySelector(".resultado");

const borrar= document.querySelector(".borrar");

const igual= document.querySelector(".igual");

Con console log podemos mostrar a través de inspeccionar de la pagina web lo de la siguiente imagen. Nos muestra con querySelectorAll muestra los números gracias al este método.





Vamos agregando funciones y manejadores de evento

numeros.forEach((num) => {

    num.addEventListener("click", () =>{

        console.log(num.textContent);

    })

Se puede ver que recorriendo los numero con foreach y el llamador de eventos addEventListener “click” y haciendo un console log del contenido al inspeccionar podemos verlos cuando hacemos click. En la parte de inspéccionar console.-

Query selector recibe el elemento que coincide con los selectores dados y agregando la palabra all recibe a un conjunto de elementos, la aplicamos para cada clase .operadores y .números

Declaramos con let variables que devuelven algo según su nombre opActual, opAnterior, resultado, operación. En este caso solo se dejarán las comillas para guardar los valores que se realizan.

let opActual = "";

let opAnterior = "";

let resultado = "";

let operacion = "";

mediante la función callback a la cual llamamos número Ingresamos un método denominado foreach para recorrer los arreglos definimos la función flecha a su vez agregamos un manejador de eventos addEventListener que se dispara cuando hacemos este evento es click

numeros.forEach((numero) => {

    numero.addEventListener ("click", () =>{

        opActual +=  numero.innerHTML;

        pantalla.innerHTML = opActual;

    })

})

cuando tomamos la referencia a la pantalla lo hacemos con el atributo inner.HTML y anexamos numero

    pantalla.innerHTML = resultado;

})

En el caso del botón borrar hacemos lo mismo para que funcione de forma individual

borrar.addEventListener("click", () =>{

    pantalla.innerHTML = "";

  opActual = "";

numeros.forEach((numero) => {

    numero.addEventListener("click", () => {

        opActual += numero.innerHTML;

        pantalla.innerHTML = opActual;

Para poder guardar el valor y la pantalla se limpie utilizamos esta seccion de código

//utilizamos cada uno de los operadores para los calculos

igual.addEventListener("click", () =>{

    switch(operacion){

        case "+":

            resultado = parseFloat(opAnterior) + parseInt(opActual);

            break;

        case "-":

            resultado = parseFloat(opAnterior) - parseInt(opActual);

            break;

        case "\*":

            resultado = parseFloat(opAnterior) \* parseInt(opActual);

            break;

        case "/":

            resultado = parseFloat(opAnterior) / parseInt(opActual);

            break;

        default:

            console.log("Operacion no valida");

    }

    pantalla.innerHTML = resultado;

})

Parseamos con parseFloat porque con parseInt nos toma un string y para evitar esto usamos el primer parse.

borrar.addEventListener("click", () =>{

    pantalla.innerHTML = "";

    opActual = "";

})

Por ultimo agregamos el botón borrar la pantalla que agregamos como

const borrar= document.querySelector(".borrar");

con el manejador de eventos lo llamamos para que ejecute la acción.—

y así concluye la calculadora con estructura HTML, formato con CSS

y acciones con JS. Realizando las operaciones de suma, resta, multiplicación y división.

