Universidad Tecnológica Metropolitana

Estructuras de Datos Aplicadas

ISC. Ruth Betsaida Martínez Domínguez, MGTI

Práctica 2

Soberanis Acosta Jimena Monserrat

Desarrollo de Software Multiplataforma

Cuarto Cuatrimestre

4°B

Parcial I

Jueves, 14 de septiembre de 2023

index.html > ...

```html
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Calculadora Matemática</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1 style="color: rgb(0, 0, 0);">Calculadora Matemática</h1>
    <div class="calculadora">
        <div class="seccion">
            <h4>Factorial</h4>
            <input type="number" id="factorialInput" placeholder="Ingrese su número">
            <button id="factorialBoton">Calcular</button>
            <p id="factorialResultado"></p>
        </div>
        <div class="seccion">
            <h4>Fabonacci</h4>
            <input type="number" id="fibonacciInput" placeholder="Ingrese su número">
            <button id="fibonacciBoton">Calcular</button>
            <p id="fibonacciResultado"></p>
        </div>
        <div class="seccion">
            <h4>Máximo Común Divisor (MCD)</h4>
            <input type="number" id="mcdInput1" placeholder="Ingrese el número 1">
            <input type="number" id="mcdInput2" placeholder="Ingrese el número 2">
            <button id="mcdBoton">Calcular</button>
            <p id="mcdResultado"></p>
        </div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Comencé con mi HTML y su estructura, coloqué 3 secciones de las respectivas prácticas, cada una tenía un input donde colocar el número, pero en el caso del Máximo Común Divisor tiene 2 por la comparación de dos números.

script.js > Calculadora

```javascript
class Calculadora{
    constructor(){
        this.factorialInput = document.getElementById("factorialInput");
        this.factorialResultado = document.getElementById("factorialResultado");
        this.fibonacciInput = document.getElementById("fibonacciInput");
        this.fibonacciResultado = document.getElementById("fibonacciResultado");
        this.mcdInput = document.getElementById("mcdInput");
        this.mcdInput1 = document.getElementById("mcdInput1");
        this.mcdInput2 = document.getElementById("mcdInput2");
        this.mcdResultado = document.getElementById("mcdResultado");
    }
```

Proseguí con hacer la parte lógica de mi aplicación, en donde comencé creando la clase Calculadora encarga de tener todos los métodos que necesitaría, aunque lo primero que puse fue crear las variables que llamarían a los inputs del HTML.

```
12
13      calcularFactorialRecursivo(n){
14          if(n === 0 || n===1){
15              return 1;
16          }else{
17              return n * this.calcularFactorialRecursivo(n-1);
18          }
19      }
20
21      calcularFactorial(n){
22          const numero = parseInt(this.factorialInput.value);
23          if (!isNaN(numero) || numero>=0){
24              const factorial = this.calcularFactorialRecursivo(numero);
25              this.factorialResultado.textContent=`El factorial del numero ${numero} es ${factorial}`;
26          }else{
27              this.factorialResultado.textContent="Por favor, ingrese un numero válido";
28          }
29      }
```

El primer método fue calcularFactorialRecursivo, que básicamente es la fórmula que seguiría para calcular la factorial de cualquier número positivo, el caso base se debe a que la factorial tanto de 0 como 1 es 1, y mientras el número que obtenga sea diferente de estos se seguirá con la misma fórmula (que resta 1 al número, y con ese mismo se multiplicaría).

Después, existe el calcularFactorial, que es la parte que interactuaría con el HTML y obtendría los parámetros, tomé en cuenta que el número debería ser de tipo número y que fue igual o mayor a 0 para proseguir con el cálculo. En este método se llamaría al otro método de Recursividad para poder sacar la factorial de número dado y de forma textual se daría el resultado.

```
30
31      calcularFibonacciRecursivo(n){
32          if(n<=1){
33              return n;
34          }else{
35              return this.calcularFibonacciRecursivo(n-1) + this.calcularFibonacciRecursivo(n-2);
36          }
37      }
38
39      calcularFibonacci(){
40          const numero = parseInt(this.fibonacciInput.value);
41          if(!isNaN(numero) && numero >=0){
42              const fibonacci = this.calcularFibonacciRecursivo(numero);
43              this.fibonacciResultado.textContent=`El número en la posición ${numero} de la serie de Fibonacci es ${fibonacci};`;
44          }else{
45              this.fibonacciResultado.textContent="Por favor, ingrese un número válido";
46          }
47      }
```

Lo mismo hice con Fibonacci, creé dos métodos, uno para la recursividad en donde el caso base si el numero sea menor o igual a 1, se daría como resultado el mismo número, sino se calcularía sumando el numero anterior del mismo número y el numero dos lugares antes y si no se sigue cumpliendo el caso base se seguiría con la recursividad.

Para el otro método calcularFibonacci comprobé que el numero que se pueda calcular sea de tipo número y mayor o igual a 0, en caso de ser así se puede seguir con devolver textualmente el resultado utilizando el método correspondiente de recursividad.

```
48
49      calcularMCDrecursivo(a, b){
50          if(b===0){
51              return a;
52          }else{
53              return this.calcularMCDrecursivo(b, a % b);
54          }
55      }
56
57      calcularMCD(){
58          const numero1 = parseInt(this.mcdInput1.value);
59          const numero2 = parseInt(this.mcdInput2.value);
60
61          if(!isNaN(numero1) && !isNaN(numero2)){
62              const mcd = this.calcularMCDrecursivo(numero1, numero2);
63              this.mcdResultado.textContent=`El Máximo Común Divisor de los números ${numero1} y ${numero2} es de ${mcd}`;
64          }else{
65              this.mcdResultado.textContent="Por favor, ingrese números válidos";
66          }
67      }
68  }
```

Para el último, también hice dos métodos, uno que fue el de calcularMCDrecursivo, en donde está la fórmula de cómo calcular el MCD, el cual tiene dos variables que se van a comparar, se tomarán dos variables a y b, se realiza la operación de a%b que significa tomar el resto de la división, si no existe tal resto, es decir, que sea exacto (0), entonces se cumpliría el caso base de b===0, que en ese caso el MCD sería a, pero sino es 0 y existe un resto en la división, se repetiría hasta que b diera 0.

Después el último método sería calcularMCD que pide los dos números que se envían en los inputs (a y b), se comprueba que ambos sean de tipo número y se utiliza el método de calcularMCDrecursivo y de manera textual nos da el resultado.

```
70  const calculadora = new Calculadora();
71
72  document.getElementById("factorialBoton").addEventListener("click", ()=>{
73      calculadora.calcularFactorial();
74  });
75
76  document.getElementById("fibonacciBoton").addEventListener("click", ()=>{
77      calculadora.calcularFibonacci();
78  });
79
80  document.getElementById("mcdBoton").addEventListener("click", ()=>{
81      const numero1 = parseInt(calculadora.mcdInput1.value);
82      const numero2 = parseInt(calculadora.mcdInput2.value);
83      calculadora.calcularMCD(numero1, numero2);
84  });
```

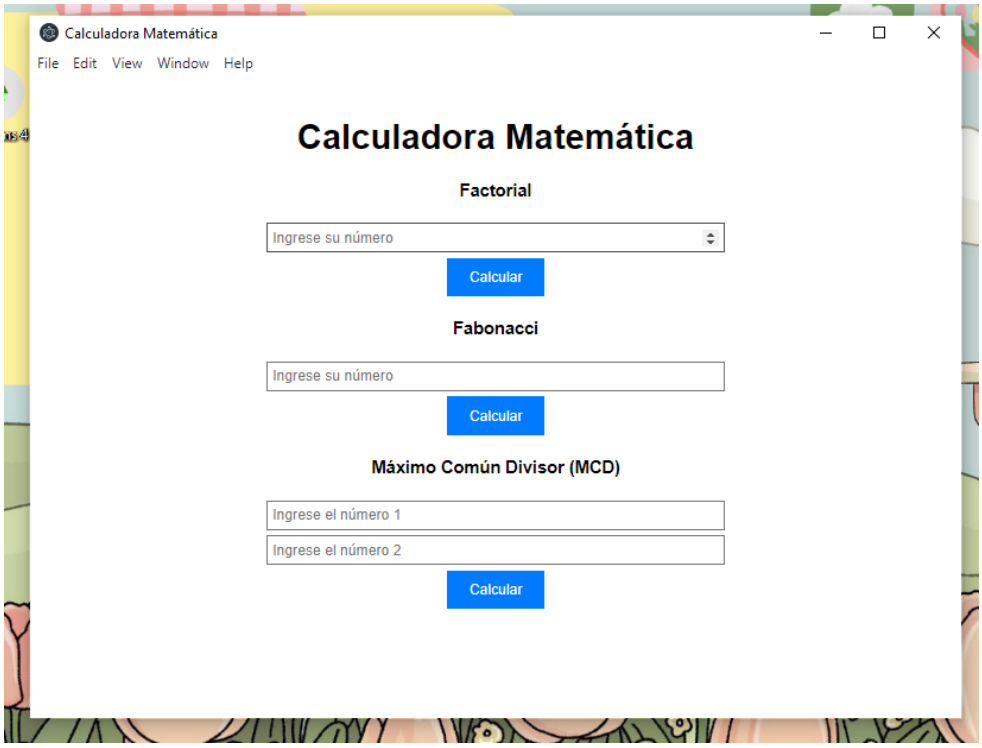
Por último, añadí un evento para cada botón de calcular después de haber creado un objeto calculadora que me ayudaría a llamar a los métodos respectivos para obtener el resultado.

Para el aspecto de la aplicación usé algo simple:

```css
body {
    font-family: Arial, sans-serif;
    text-align: center;
    padding: 10px;
}

h1 {
    color: #333;
}

.calculatora {
    display: flex;
    justify-content: space-between;
    gap: 10px;
}

.section {
    border: 1px solid #ccc;
    padding: 10px;
}

input[type="number"] {
    width: 50%;
    padding: 5px;
    margin-bottom: 5px;
}
```

```css
button {
    background-color: #007bff;
    color: white;
    border: none;
    padding: 10px 20px;
    cursor: pointer;
    display: block;
    margin: auto;
}

button:hover {
    background-color: #0056b3;
}

p {
    margin-top: 5px;
    font-weight: bold;
}
```

Para crear la aplicación volví a utilizar el framework Electron, y al final el aspecto quedó así:



Ejemplo de uso: