Universidad Tecnológica Metropolitana

Estructuras de Datos Aplicadas

ISC. Ruth Betsaida Martínez Domínguez, MGTI

Práctica 5-6

Soberanis Acosta Jimena Monserrat

Desarrollo de Software Multiplataforma

Cuarto Cuatrimestre

4°B

Parcial I

Viernes, 15 de septiembre de 2023

PRÁCTICA 5

index.html > ...

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Cambio de Monedas</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Cálculo de Cambio</h1>
        <label for="montoTotal">Total de la Compra:</label>
        <input type="number" id="montoTotal">
        <label for="totalPagar">Monto de pago:</label>
        <input type="number" id="totalPagar">
        <div class="button-container">
            <button id="calcular">Calcular Cambio</button>
        </div>
        <div id="resultado"></div>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

Comencé con la estructura principal del HTML en donde solo hay dos inputs del Total de la compra y el Monto de pago, y por último un botón para poder calcular el cambio.

script.js > CambioCalculator > calcularCambio

```js
class CambioCalculator {
    constructor() {
        this.denominaciones = [100, 50, 20, 10, 5, 1, 0.5, 0.2, 0.01];
    }

    calcularCambio() {
        const montoTotal = parseFloat(document.getElementById('montoTotal').value);
        const totalPagar = parseFloat(document.getElementById('totalPagar').value);

        if (isNaN(montoTotal) || isNaN(totalPagar)) {
            alert("Por favor, ingresa montos válidos.");
            return;
        }
```

Luego, proseguí con el script, en donde comencé con una clase llamada CambioCalculadora, el cual tiene un constructor con un arreglo de denominaciones o también los pesos que se puede tomar.

Después hice el primer método llamado CalcularCambio() en donde hay dos nuevas const que guardan el monto total y el pago que se hará, también hice la comprobación de ingresar montos válidos.

```
        const cambioTotal = totalPagar - montoTotal;
        let cambioRestante = cambioTotal;
        const resultado = {};

        for (const denominacion of this.denominaciones) {
            const recuentoMonedas = Math.floor(cambioRestante / denominacion);
            if (recuentoMonedas > 0) {
                resultado[denominacion] = recuentoMonedas;
                cambioRestante -= recuentoMonedas * denominacion;
                cambioRestante = parseFloat(cambioRestante.toFixed(2));
            }
        }

        this.displayResult(resultado);
    }

```

Proseguí creando un bucle que lo que hace es recorrer entre el arreglo de denominaciones que hice antes. Entonces creé la constante de recuentosMonedas en donde calcula cuántas monedas de la denominación se necesita para el cambio y utiliza el Math.floor para tomar solo el resultado que sea número entero.

Entonces con el if si ese recuento es mayor a 0 puede calcular el resultado. El resultado[denominacion] = recuentoMonedas lo que hace es si se deben dar monedas o billetes de esa denominación, se registra la cantidad en el objeto resultado, utilizando la denominación como clave y la cantidad como valor. Después se actualiza el valor de cambio restante restando a la cantidad de denominación actual que dio como cambio.

La última línea y utiliza para limitar la cantidad de decimales en cambioRestante a dos decimales, ya que estamos trabajando con valores de dinero. Esto ayuda a evitar problemas de redondeo en cálculos con decimales.

```
        this.displayResult(resultado);
    }

    displayResult(resultado) {
        const resultElement = document.getElementById('resultado');
        resultElement.innerHTML = 'Cambio a devolver:<br>';

        for (const denominacion in resultado) {
            resultElement.innerHTML += `${resultado[denominacion]} moneda de ${denominacion} pesos<br>`;
        }
    }
}

```

Proseguí usando el método displayResult que básicamente muestra el resultado en un formato específico.

Ese displayResult crea el const resultElemente que llama desde el HTML el id resultado, entonces cuando lo haga se modifica el HTML de ese lugar con el texto que de muestra.

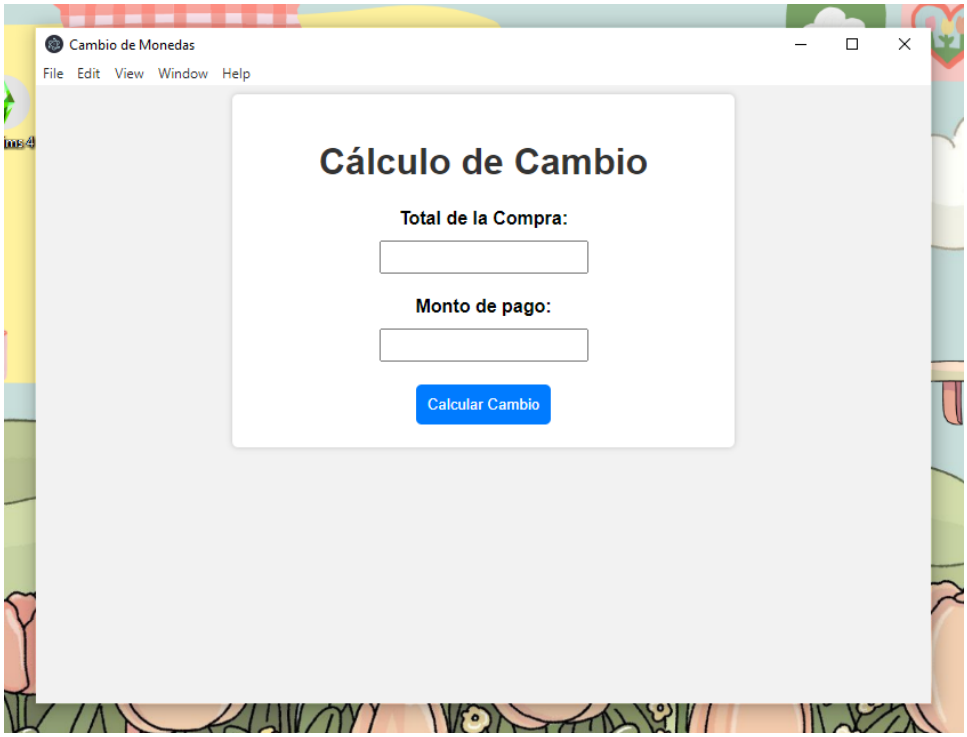
Después en ese método se crea un bucle que recorre las denominaciones en el objeto resultado, entonces por cada uno se imprime en el HTML el texto que se ve y al final salta a la siguiente línea con el

Por ultimo se crea una instancia de la clase CambioCalculadora, después se busca el elemento con el id calcular, en donde al momento de hacer click se ejecutará una función anónima que llamará al método calcularCambio y así comenzar a realizar los cambios.

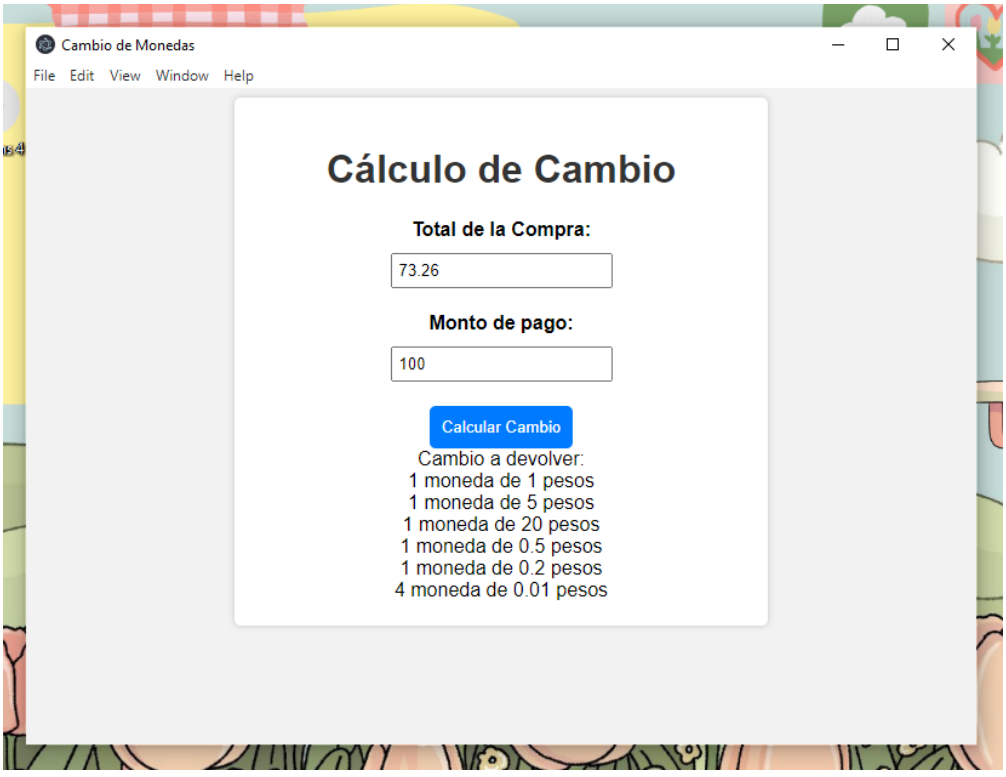
```
# styles.css > ...
body {
    font-family: Arial, sans-serif;
    background-color: #f2f2f2;
}

.container {
    max-width: 400px;
    margin: 0 auto;
    padding: 20px;
    background-color: #fff;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.2);
    border-radius: 5px;
    text-align: center;
}

h1 {
    color: #333;
}

label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}

input {
    width: auto;
    padding: 5px;
    margin-bottom: 20px;
}
```

```
.button-container{
    display: flex;
    justify-content: center;
}

button {
    background-color: #007BFF;
    color: #fff;
    border: none;
    padding: 10px;
    border-radius: 5px;
    cursor: pointer;
}

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

#result {
    margin-top: 20px;
    font-weight: bold;
}
```

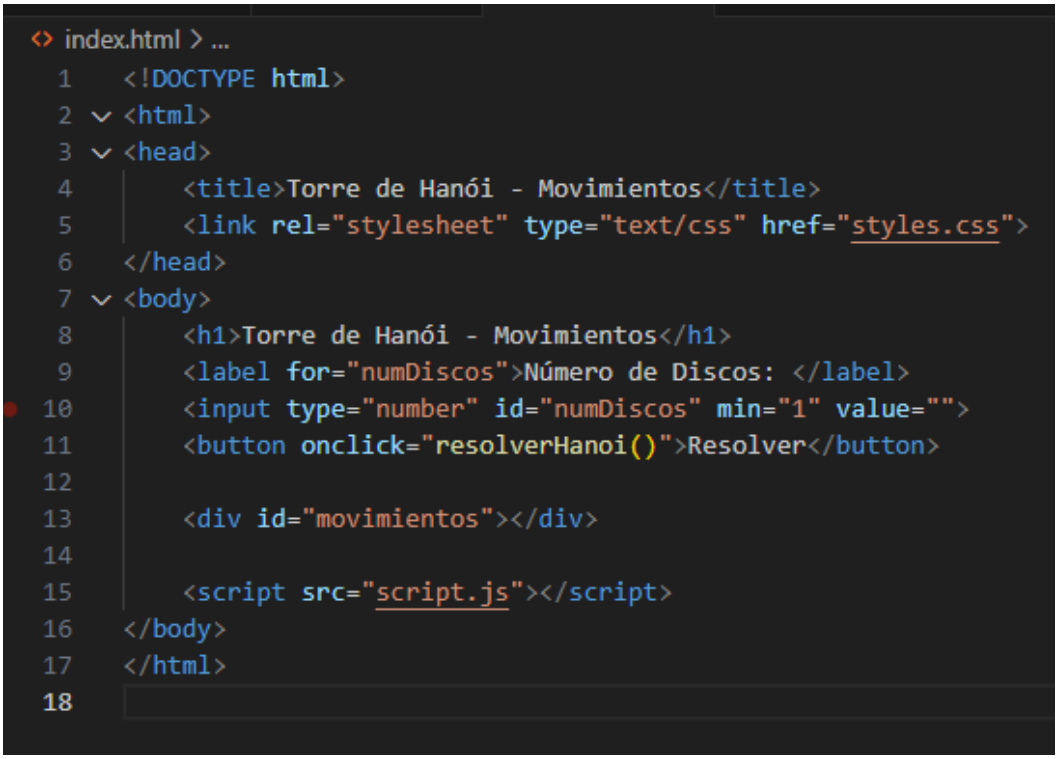
Terminé agregando el estilo del HTML, y con ayuda del framework Electron lancé la aplicación y así quedó:



Y usando el ejemplo de la práctica, con el total de compra de 73.26 y pagando con 100, nos daría:



PRÁCTICA 6



```html
<!DOCTYPE html>
<html>
<head>
    <title>Torre de Hanói - Movimientos</title>
    <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
    <h1>Torre de Hanói - Movimientos</h1>
    <label for="numDiscos">Número de Discos: </label>
    <input type="number" id="numDiscos" min="1" value="">
    <button onclick="resolverHanoi()">Resolver</button>

    <div id="movimientos"></div>

    <script src="script.js"></script>
</body>
</html>
```

Comencé con la estructura del HTML donde solo necesitaba un input para poner el número de discos y un botón que resuelve el problema, pues resolverHanoi es una función que se activa cuando se hace algún click en la página, este mismo se explicará más adelante.

```javascript
class TorreHanoi {
    constructor() {
        this.movimientos = [];
    }

    resolver(numDiscos, origen, auxiliar, destino) {
        if (numDiscos === 1) {
            this.movimientos.push(`Mover disco de la torre ${origen} a la torre ${destino}`);
        } else {
            this.resolver(numDiscos - 1, origen, destino, auxiliar);
            this.movimientos.push(`Mover disco de la torre ${origen} a la torre ${destino}`);
            this.resolver(numDiscos - 1, auxiliar, origen, destino);
        }
    }
}
```

Por la parte del script, empecé con una clase llamada TorreHanoi que tiene un constructor con un arreglo vacío llamado movimientos, en donde estos mismos se guardarán ahí.

Después creé el método resolver el cual tiene de parámetros, el número de discos, la torre origen, la torre auxiliar y la torre de destino.

Con la condición if en donde si el número de discos es igual a 1, se ordena que se envíe el texto que aparece ahí, pues si solo hay 1 disco, solo hay una sola solución que es de origen a destino, en este ejercicio representa el caso base de la recursión.

Entonces, si hay más de dos discos, se realiza el método llamado resolver en dos ocasiones:

1. this.resolver(numDiscos - 1, origen, destino, auxiliar); en donde al número de discos -1 toma el disco de la torre de origen a la torre de nombre auxiliar, utilizando a la torre de destino como ayuda.
2. Es ahí se registra el movimiento del último disco restante de la torre de origen a la torre de destino.
3. this.resolver(numDiscos - 1, auxiliar, origen, destino); en donde finalmente, se mueven los numDiscos - 1 discos de la torre auxiliar a la torre de destino, utilizando la torre origen como torre de ayuda.

Este programa utiliza la recursión para mover los discos correctamente. Primero, se mueven los discos más pequeños a la torre auxiliar, luego se mueve el disco más grande a la torre destino, y finalmente se mueven los discos más pequeños a la torre destino utilizando la torre origen como ayuda.

```
16
17  function resolverHanoi() {
18      const numDiscos = parseInt(document.getElementById('numDiscos').value);
19      const torre = new TorreHanoi();
20      torre.resolver(numDiscos, '1', '2', '3');
21      const movimientosDiv = document.getElementById('movimientos');
22      movimientosDiv.innerHTML = torre.movimientos.join('<br>');
23  }
24
```

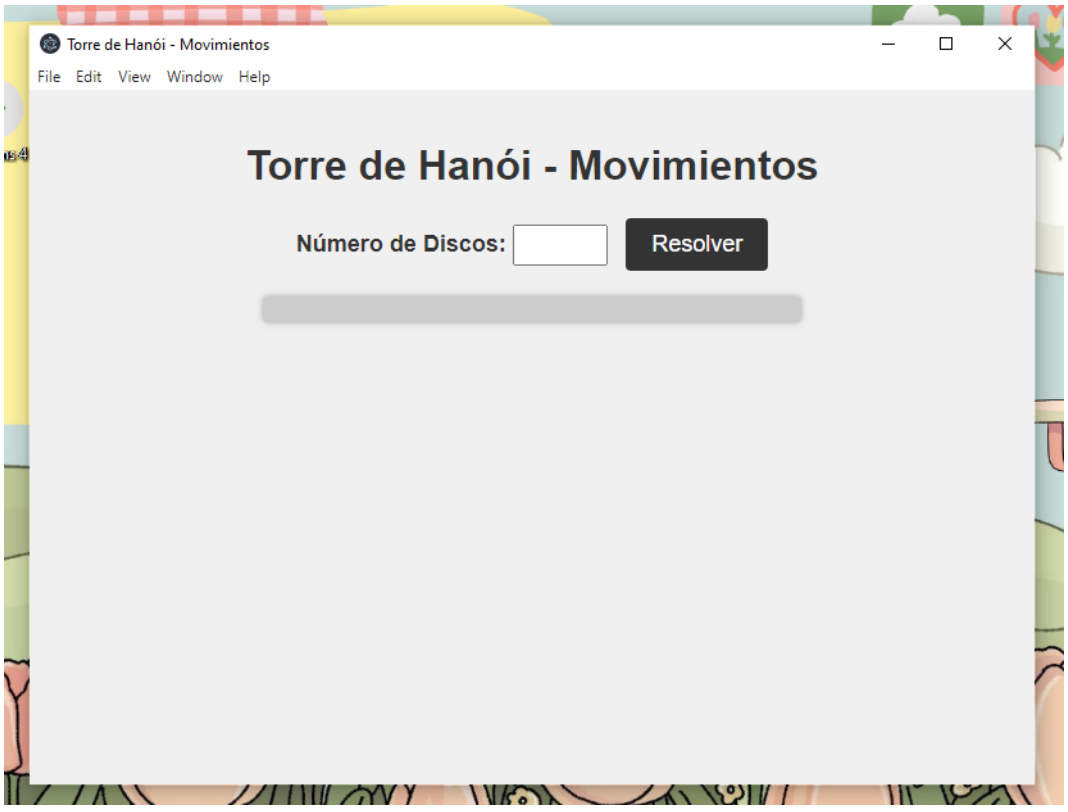
Por último, en este script, creé una función llamada resolverHanoi que se pude ver en el HTML pues se activa cuando se hace click en el botón, este comience creando un variable que obtiene el número de disco que se haya ingresado en un campo de entrada de la aplicación, obtiene el contenido que tiene en esa entrada, el cual debería ser un número y lo convierte en un entero que se almacena en esa variable llamada numDiscos.

Se crea una nueva instancia del objeto TorreHanoi que se guarda en la variable torre. También se guarda en la variable movimientosDiv el contenido que haya en el elemento con el id movimientos. Y esta misma variable actualiza el contenido del HTML. Torre.movimientos es un arreglo que almacena los movimientos hechos en una cadena de texto, el cual lo une el join y junto al
 hace que haya un salto de línea, para que no sea seguida la cadena.

Esta función muestra al final los movimientos realizados.

```css
# styles.css > body
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
    text-align: center;
    padding: 20px;
    margin: 0;
}

h1 {
    color: #333;
}

label {
    font-size: 18px;
    font-weight: bold;
    color: #333;
}

input[type="number"] {
    width: 60px;
    padding: 5px;
    font-size: 16px;
}

button {
    background-color: #333;
    color: #fff;
    border: none;
    padding: 10px 20px;
    font-size: 18px;
    cursor: pointer;
    border-radius: 4px;
    margin-left: 10px;
}

button:hover {
    background-color: #555;
}

#movimientos {
    text-align: left;
    margin-top: 20px;
    font-size: 19px;
    padding: 10px;
    background-color: #ccc;
    border-radius: 4px;
    box-shadow: 0 0 5px rgba(0, 0, 0, 0.3);
    max-width: 400px;
    margin-left: auto;
    margin-right: auto;
}
```

Por último, con el css le di el estilo que quería a la aplicación y con ayuda del framework de Electron pude lanzar la misma, y así quedó:



Y usando de ejemplo de usar 2 discos:

Y también usando 3 discos: