Aplicando Clean Code

Separacion de Funciones

La separación de funciones en diferentes archivos o en un solo archivo es una cuestión de diseño y puede depender del tamaño y complejidad del proyecto, así como de las preferencias.

Sin embargo, siguiendo los principios de Clean Code, es recomendable que cada archivo contenga una sola responsabilidad o concepto, lo que se conoce como el principio de responsabilidad única (SRP). Esto significa que cada archivo debe tener un propósito claro y específico, y contener solo el código necesario para cumplir con esa responsabilidad.

Veamos las funciones que a criterio personal se desglosaron:
procesarNumero.js procesarNumero.js
numeroValido.js numeroValido.js
numeroValido.js calcularSiguienteNumero.js
procesarNumero.js calcularSaltos.js
numeroValido.js mostrarTabla.js
numeroValido.js mostrarMensaje.js

procesarNumeros.js

Descripcion:
Con esta funcion se empleana lgunos principios importantes de código limpio, como el uso de nombres significativos, la validación de entrada de usuario y la modularización del código.

JS procesarNumero.js

```js
function procesarNumero() {
    const numero = parseInt(document.getElementById("numero").value);

    if (esNumeroValido(numero)) {
      const saltos = calcularSaltos(numero);
      mostrarTabla(saltos);
    } else {
      mostrarMensaje("El número debe ser mayor que 10000.");
    }
  }

```

numeroValido.js

Descripcion:
Con esta función se verifica si el número es mayor que 10000

JS numeroValido.js

```js
// Verifica si el número es mayor que 10000
function esNumeroValido(numero) {
    return numero > 10000;
  }
```

calcularSaltos.js

Descripcion:
Calcula la trayectoria de la nave y cuenta los saltos realizados, esta funcion sigue algunos principios importantes de código limpio, como el uso de nombres significativos, el uso de constantes y ciclos while para procesar tareas repetitivas y el uso de comentarios para explicar el código.

JS calcularSaltos.js

```js
// Calcula la trayectoria de la nave y cuenta los saltos realizados
function calcularSaltos(numero) {
    let saltos = 0;
    const trayectoria = [];

    while (numero > 1) {
      saltos++;
      trayectoria.push(numero);
      numero = calcularSiguienteNumero(numero);
    }

    trayectoria.push(numero);
    trayectoria.push(`Se han dado ${saltos} saltos.`);
    trayectoria.push("La trayectoria de su nave ha sido trazada.");

    return trayectoria;
  }
```

calcularSiguienteNumero.js

Descripcion:
Calcula el siguiente número en la secuencia sigue algunos principios importantes de código limpio, como el uso de nombres significativos, el uso de operadores ternarios para simplificar el código y el uso de espacios y sangría para mejorar la legibilidad.

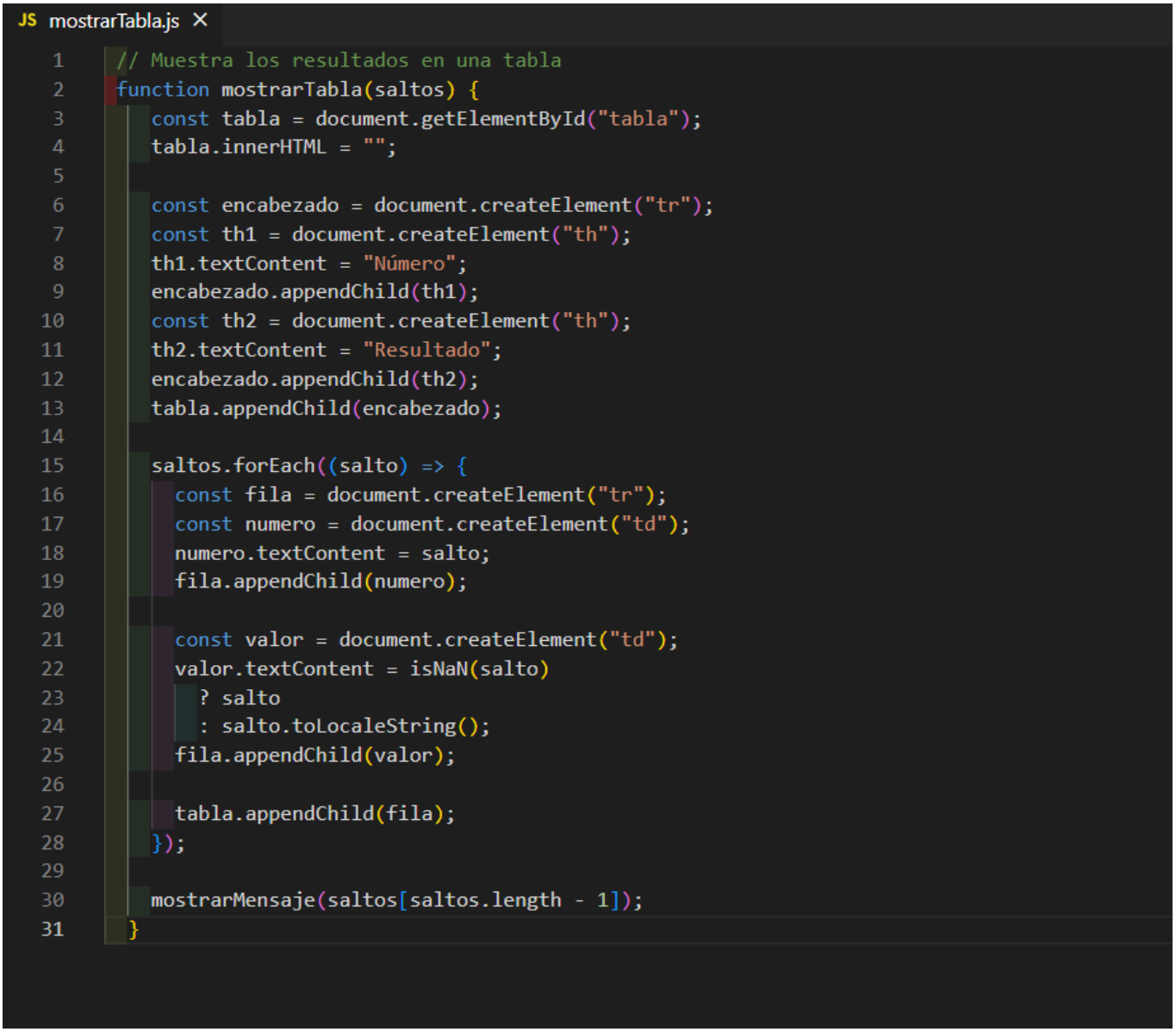
JS calcularsiguienteNumero.js

```js
// Calcula el siguiente número en la secuencia
function calcularSiguienteNumero(numero) {
    return numero % 2 === 0 ? numero / 2 : 3 * numero + 1;
  }
```

mostrarTabla.js

Descripcion:
Sigue algunos principios importantes de código limpio, como el uso de nombres significativos, el uso de constantes, createElement() para crear elementos HTML, forEach() para procesar tareas repetitivas, isNaN() para comprobar si un valor es un

número, y mostrarMensaje() para mostrar un mensaje en el DOM.



mostrarMensaje.js

Descripción:
Sigue algunos principios importantes de código limpio, como el uso de nombres significativos, el uso de constantes, y el uso de className para establecer una clase CSS.
// Muestra un mensaje en pantalla

JS mostrarMensaje.js

```
1
2   // Muestra un mensaje en pantalla
3   function mostrarMensaje(mensaje) {
4     const divMensaje = document.getElementById("mensaje");
5     divMensaje.textContent = mensaje;
6     divMensaje.className = "mensaje";
7   }
```

limpiar.js

JS limpiar.js

```
1   function limpiar() {
2     const tabla = document.getElementById("tabla");
3     tabla.innerHTML = "";
4     const divMensaje = document.getElementById("mensaje");
5     divMensaje.textContent = "";
6   }
```

index.html

index.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Asignación Uno // Módulo II Preespecialización en la Nube</title>
    <link rel="stylesheet" href="style.css">

</head>
<body>
    <h1> Planeta Namek</h1>

        <form>
            <label for="numero">Número para operar nave:</label>
            <input type="number" id="numero" name="numero" min="1" max="100000" />
            <button type="button" onclick="procesarNumero()">Validar</button>
        </form>
        <form>
            <button id="limpiarBtn">Limpiar</button>

        </form>
        <br>

        <div id="tabla">

        </div>

        <table id="tabla">
        </table>
        </div>
        <div id="mensaje"></div>
</body>

    <div id="mensaje"></div>

<script src="calcularSaltos.js"></script>
<script src="calcularsiguienteNumero.js"></script>
<script src="mostrarMensaje.js"></script>
<script src="mostrarTabla.js"></script>
<script src="numeroValido.js"></script>
<script src="procesarNumero.js"></script>
<script src="limpiar.js"></script>
</html>

```

resultado

Planeta Namek

Número para operar nave: 10500 Validar

Limpiar

Número	Resultado
10500	10,500
5250	5,250
2625	2,625
7876	7,876
3938	3,938
1969	1,969
5908	5,908
2954	2,954
1477	1,477
4432	4,432
2216	2,216
1108	1,108
554	554
277	277
832	832
416	416
208	208
104	104
52	52
26	26
13	13
40	40
20	20
10	10
5	5
16	16
8	8
4	4
2	2
1	1
Se han dado 29 saltos.	Se han dado 29 saltos.
La trayectoria de su nave ha sido trazada.	La trayectoria de su nave ha sido trazada.

La trayectoria de su nave ha sido trazada.