

CUADERNO PERSONAL

DESARROLLO DE APLICACIONES WEB

SEMANA 4 – DESARROLLO FRONTEND CON ECMAScript



- JavaScript: Es un lenguaje de programación script dinámico, multi-paradigma, basado en prototipos, dinámicos, soporta estilos de programación funcional, orientada a objetos e imperativa, además ahora JavaScript no solamente en un lenguaje para web, sino que te permite crear:
 - a) Aplicaciones de servidor
 - b) Aplicaciones móviles
 - c) Aplicaciones de escritorio
 - d) Aplicaciones de consola
 - e) Aplicaciones de robótica
 - f) Aplicaciones empotradas
- Node JS: Es un entorno de ejecución para JavaScript construido con el motor de JavaScript V8 de Chrome, al contrario que la mayoría de código JavaScript, no se ejecuta en un navegador, sino en un servidor.
- Conceptos básicos: Javascript esta influenciado por la sintaxis de Java, Awk, Perl y Python, en JS las instrucciones son llamadas **sentencias** y son **separadas por un punto y coma (;)**. Es recomendable siempre terminar con punto y coma.
- ¿Dónde se puede incluir JS?
 - a) Incluir JS en el mismo documento HTML

```
<script>  
  Codigo Java Script  
</script>
```

- b) Definir JS en un archivo externo

```
<script src="codigo.js"></script>
```

Archivo: código.js

```
/*Colocar código java script/*
```

c) Incluir JS en tag

```
<body>  
<p onclick="alert('Mensaje de prueba')">Párrafo Vinculo </p>  
</body>
```

- Variables y Operadores

- a) Variables: Son espacios de memoria con valores y tienen un nombre. El nombre de una variable tiene que empezar con una letra, un guion bajo (_) o un símbolo de dólar.

Variable global o local (Depende donde es declarado)

Sintaxis:

```
var Nom_Variable;
```

O

```
var Una_Variable = "Texto";
```

Variable local

Sintaxis:

```
let Nom_Variable;
```

O

```
let Una_Variable = "Texto";
```

Constante

Usar const en:

- Una nueva matriz
- Un nuevo objeto
- Una nueva función
- Una nueva expresión regular

Sintaxis:

```
const Nom_constante;
```

O

```
const Nom_constante = "Texto";
```

- Variables globales: Son propiedades del objeto global. En las páginas web, el objeto global es window, se puede establecer y acceder a las variables globales usando la sintaxis window.variable. Podemos acceder a variables globales declaradas en una ventana o frame de otra ventana o frame especificando el nombre de la ventana o frame.

- Tipos de datos: Según ECMAScript existen 8 tipos de datos: Boolean, Null, Undefined, Number, BigInt, String, Symbol y Object

Boolean

Este tipo de dato almacena un bit que indica true o false.

```
var si = true; var no = false;
```

null

Valor nulo. null

```
var x = null;
```

undefined.

Una propiedad de alto nivel cuyo valor no es definido.

```
var dato; // su valor es undefined var dato = undefined;
```

Number

Un número entero o un número con coma flotante.

```
var miEntero = 1; var miDecimal = 1.33;
```

BigInt

Un número entero con precisión arbitraria.

```
var miEntero = 9007199254740992;
```

String

Las variables de tipo string almacenan caracteres o palabras.

```
var dato = "Esto es un string";  
var otroDato = 'Esto es otro string';
```

Symbol

Permite obtener valores que no pueden volver a ser creados, es decir, son identificadores únicos e inmutables.

```
const myFirstSymbol = Symbol();
```

- Object: Un objeto es una estructura de datos complejos en memoria identificada con un nombre. Es una colección de propiedades. Cuando se declara un objeto, un limitado grupo de propiedades son inicializadas; luego pueden ser agregados o eliminadas otras propiedades; las propiedades se identifican usando claves. Una clave es un valor String o Symbol.
- Tipos de propiedades de objetos:
 - a) Propiedad de datos

Atributo	Tipo	Descripción	Valor por defecto
[[Value]]	Cualquier tipo JavaScript	El valor obtenido mediante un acceso get a la propiedad.	undefined
[[Writable]]	Boolean	Si es false, el [[Value]] de la propiedad no puede ser cambiado.	false
[[Enumerable]]	Boolean	Si es true, la propiedad será enumerada en ciclos for...in . Consultar también Enumerabilidad y pertenencia de las propiedades	false
[[Configurable]]	Boolean	Si es false, la propiedad no puede ser eliminada y otros atributos que no sean [[Value]] y [[Writable]] no pueden ser cambiados.	false

b) Propiedad de acceso

Atributo	Tipo	Descripción	Valor por defecto
[[Get]]	Función, objeto o undefined	La función es llamada con una lista de argumentos vacía y devuelve el valor de la propiedad cada vez que un acceso get al valor es realizado. Consultar también get .	undefined
[[Set]]	Function object or undefined	La función es llamada con un argumento que contiene la variable asignada y es ejecutada siempre que una propiedad específica se intenta cambiar. Consultar también set .	undefined
[[Enumerable]]	Boolean	Si el valor es true, la propiedad será enumerada en bucles for...in.	false
[[Configurable]]	Boolean	Si el valor es false, la propiedad no podrá ser eliminada y tampoco ser modificada a una propiedad de datos.	false

- Variable hoisting: Las variables en JS pueden hacer referencia a una variable declarada más tarde, sin obtener una excepción, este concepto se conoce como hoisting, las variables en JS son en cierto sentido “elevadas” a la parte superior de la función o declaración.
- Operadores: Son símbolos que denotan operaciones que pueden realizarse entre elementos llamados operandos.

Aritmeticos						
+	-	*	/	%	++	--

Cadena		
+	Concatenacion.	Nota: "cadena "+15 el resultado es "cadena 15" . "37"-7 resultado 30 "37"+7 resultado "377"

Lógicos					
&&	Y		O	!	Negacion

Comparación			
>		>=	
<		<=	
==	Igualdad Devuelve true si ambos operadorandos son iguales.	===	Estrictamente iguales Devuelve true si los operandos son iguales y tienen el mismo tipo.
!=	Desigualdad	(!==)	Estrictamente desiguales

Asignación					
Operador	Significado	Operador	Significado	Operador	Significado
x = y	x = y	x *= y	x = x * y	x %= y	x = x % y
x += y	x = x + y	x /= y	x = x / y	x **= y	x = x ** y
x -= y	x = x - y				

Condicional (ternario)	
Condición ? valor1: valor2	Si la condición es true, devuelve el valor1, de lo contrario el valor2.
Ejemplo	var estado = (edad >= 18) ? "adulto" : "menor";

- Operadores Spread: Se representan con tres puntos: ...; es operador spread permite expandir elementos iterables, como, arrays, cadenas de texto, objetos, listas de nodos, el objeto arguments de una función y otros dos tipos de estructuras nuevas en ES6: maps y sets.

```
//Dado el siguiente array:
const alfabeto = ["a","b","c","d","e"];

//Si lo sacamos expandido en consola:
console.log(...alfabeto);// a b c d e f
```

- Estructuras de control
 - a) El condicional if

```
if(condicion) {  
    codigo necesario }  
else {  
    codigo alternativo }
```

b) El condicional switch

```
switch(opcion) {  
    case caso1 :  
        sentencias para caso1;  
        break;  
        .....  
    case casoN :  
        sentencias para casoN;  
        break;  
    default :  
        sentencias por defecto;  
        break;  
}
```

- Estructuras repetitivas o bucles

a) Bucle for

```
for([Inicialización]; [Condición]; [Expresión de actualización]){  
    Instrucciones a repetir  
}
```

b) Bucle while

```
while(Condición) {  
    Instrucciones a repetir  
}
```

c) Bucle Do While

```
do {  
    Instrucciones a repetir  
} while(condicion);
```

```
var text = ""  
var i = 0;  
do {  
    text += "<br>El número es: " + i;  
    i++;  
}while (i < 5);  
document.getElementById("demo").innerHTML = text;  
}
```

d) Foreach

```
Nom_array.forEach(nomFuncion, thisValue);  
function nomFuncion(elemento, indice, arr){  
   Codigo  
};
```

e) Bucle For/Of

```
for (variable of iterable) {  
    Sentencias  
}
```

f) Bucle For/In

```
for( indice in objeto )
```

- Funciones y Procedimientos

a) Funtion Declaration

```
function NombreFuncion(param1, ..., paramN){  
    Código de la función  
  
    return Valor;  
}
```

b) Function Expression o Funciones Anónimas

¿Qué sucede si la función se asigna a una variable?

```
var x = function nomFuncion(a, b) { return a * b; }
```

Para llamar a función utilizo: x(3,5) y ya no puedo usar nomFuncion(3,5)

Función anónima

```
var x = function (a, b) {return a * b};  
var z = x(4, 3);
```

- Funciones y procedimientos: Una función de JavaScript es un bloque de código diseñado para realizar una tarea en particular y se ejecuta cuando lo invocan (o lo llaman).

- a) Función Declaration: Se define con la palabra reservada function, seguida de nombre, paréntesis y bloque de código.

```
function NombreFuncion(param1, ..., paramN){  
    Código de la función  
  
    return Valor;  
}
```

Llamar una función desde javascript

```
Var f1 = NombreFuncion(arg1,...);
```

Llamar una función desde un evento de html

Podremos llamar a una función (definida en <head> del HTML) con cualquier tag, por ejemplo:

```
<p onClick="NombreFuncion(arg1,...);">Hacer Click Aqui</p>
```

- b) Function Expression o Funciones Anónimas: Son funciones que no tendrá un nombre, pero deben ser almacenadas en una variable.

¿Qué sucede si la función se asigna a una variable?

```
var x = function nomFuncion(a, b) { return a * b; }
```

Para llamar a función utilizo: x(3,5) y ya no puedo usar nomFuncion(3,5)

Función anónima

```
var x = function (a, b) {return a * b};  
var z = x(4, 3);
```

- c) Function Object: También se pueden definir con un constructor de funciones de JavaScript incorporado llamado Function().


```
var myFunction = new Function("a", "b", "return a * b");  
var x = myFunction(4, 3);
```

- d) Funciones Auto Invocadas: Es una función que se inicia automáticamente, sin ser llamada. Para auto invocar una función utilizamos el operador ().

```
(Función)(Parámetros);
```

```
(function () { //codigo })(param1, param2, ... paramN);
```

Se debe agregar paréntesis alrededor de la función para indicar que es una expresión de función:

```
Ejemplo  
(function () {  
  var x = "Hola!!"; // Función auto invocada  
})();
```

- e) Funciones de Flecha: Son funciones que trata de simplificar el código. La sintaxis que utiliza es muy corta y generalmente serán utilizadas para funciones que contengan poco código.

```
const x = function(x, y) { return x * y; }
```

```
const x = (param1, ...) => { sentencias; }
```

```
const prod = (x, y) => {  
  let p = x * y;  
  return p  
};
```

- El Objeto Arguments: Las funciones de JS tienen un objeto incorporado llamado arguments que contiene la lista de argumentos de la función.
 - a) Argumentos de la función

```
// Declaración
function NombreFuncion(param1, ..., paramN){
    Código de la función

    return Valor;
}

// Invocación
NombreFuncion(argumento1, argumento2, ..., argumentoM);
```

```
// Declaramos una función
function unaFuncion(){
    console.log(arguments);
}
// Invocamos a la función
unaFuncion(1, 2, 3);

// Resultado
// [Arguments] { '0': 1, '1': 2, '2': 3}
```

- b) Parámetros de la función: Son las variables locales listadas en la declaración de la función y pueden tomar un valor por defecto.

```
// Función con valor por defect ES6
function unaFuncion(a=valorDefecto1, b=valorDefecto2){ codigo; }
```

```
// Declaramos una función
function unaFuncion(a, b, c){
    console.log(arguments);
}
// Invocamos a la función
unaFuncion(2, 3);

// Resultado
// [Arguments] { '0': 2, '1': 3 }
```

```
// Declaramos una función
function unaFuncion(a, b){
    console.log(arguments);
}
// Invocamos a la función
unaFuncion(2, 3, 4, 5);

// Resultado
// [Arguments] { '0': 2, '1': 3, '2': 4, '2': 5 }
```

- Cadenas (String): Son textos que podemos usar en JS el tipo de dato es String. Es un conjunto de elementos de valores enteros de 16-bits no signados. Cada elemento ocupa una posición.
- a) Literales de cadena

```
'foo'
"bar"
```

b) Secuencia de escape hexadecimal

El número después de `\x` es interpretado como un número hexadecimal.

```
'\xA9' // "9"
```

c) Secuencia de escape Unicode

Las secuencias de escape Unicode requieren al menos cuatro dígitos hexadecimales después de `\u`.

```
'\u00A9' // "9"
```

d) Objetos cadena

El objeto `String` es un envoltorio alrededor del tipo de datos de cadena original.

```
var s = new String("foo"); // Crea un objeto String
console.log(s); // Muestra: { '0': 'f', '1': 'o', '2': 'o' }
typeof s; // Devuelve 'object'
```

e) Métodos String

Método	Descripción
<code>charAt</code> , <code>charCodeAt</code> , <code>codePointAt</code>	Devuelve el carácter o el código del carácter en la posición especificada en la cadena.
<code>indexOf</code> , <code>lastIndexOf</code>	Devuelve la posición de la subcadena en la cadena o la última posición de una subcadena especificada respectivamente.
<code>startsWith</code> , <code>endsWith</code> , <code>includes</code>	Devuelve si la cadena empieza, termina o contiene una cadena especificada, o no.
<code>concat</code>	Combina el texto de dos cadenas y retorna una nueva cadena.
<code>fromCharCode</code> , <code>fromCodePoint</code>	Construye una cadena desde la secuencia de valores Unicode especificada. Este es un método de la clase <code>String</code> , no una instancia de <code>String</code> .
<code>split</code>	Divide un objeto <code>String</code> en un array de strings separados por substrings.
<code>slice</code>	Extrae una sección de un string y devuelve un nuevo string.
<code>substring</code> , <code>substr</code>	Devuelve un substring del string, bien especificando el comienzo y el final, o bien el índice inicial y la longitud del substring.
<code>match</code> , <code>replace</code> , <code>search</code>	Para trabajar con expresiones regulares
<code>toLowerCase</code> , <code>toUpperCase</code>	Devuelve el string en mayúsculas o en minúsculas
<code>normalize</code>	Devuelve el string Normalizado en Unicode.
<code>repeat</code>	Devuelve un string formado por los elementos del objetos repetidos el número de veces que le indiquemos.
<code>trim</code>	Elimina los espacios en blanco del principio y del final del string.

f) Cadenas de plantillas multilinea

Se utiliza el carácter (```) (acento grave) en lugar de las comillas dobles o simples.

```
(` `) // Código ASCII Alt+96
```

Ejemplo

```
var nombre="Jaime";
console.log(`Bienvenido: ${nombre}`);
```

- Arrays, Number y Modulos JS
 - a) Number: Es la clase del tipo primitivo number

Función	Descripción	Ejemplo
<code>parseInt(string, [b])</code>	Convierte un string a número decimal Si agrega base, convierte en base b.	<code>parseInt("1111");</code> // Devuelve 1111
<code>parseFloat(string, [b])</code>	Convierte un string a número float Si agrega base, convierte en base b.	<code>parseFloat("5e3");</code> // Devuelve 5000
<code>number.toFixed(x)</code>	Redondea number con X decimales y lo convierte a string.	<code>var n = 2.5674;</code> <code>n.toFixed(0);</code> // Devuelve "3"
<code>number.toExponential(x)</code>	Redondea number float con X decimales y lo convierte a string.	<code>var n = 2.5674;</code> <code>n.toExponential(2);</code> // Devuelve "2.56e+0"
<code>number.toString(b)</code>	Convierte number a un string con base b	<code>(15).toString(2);</code> // Devuelve "1111"

- b) Modulo Math: Es una clase propia de JS que contiene valores y funciones que nos permiten realizar operaciones matemáticas.

Funcion Matematicas	Valor devuelto
<code>Math.PI;</code>	// Número Pi = 3.14159265...
<code>Math.E;</code>	// Número e = 2.7182818...
<code>Math.random();</code>	// Número aleatorio entre 0 y 1, ej: 0.45673858
<code>Math.pow(2,6);</code>	// Potencia de 2 elevado a 6 = 64;
<code>Math.sqrt(4);</code>	// raíz cuadrada de 4 = 2
<code>Math.min(4,3,1);</code>	// Devuelve el numero mínimo = 1
<code>Math.max(4,3,1);</code>	// Devuelve el numero máximo = 4
<code>Math.floor(6.4);</code>	// Devuelve el entero más próxima por debajo,
<code>Math.ceil(6.4);</code>	// Devuelve el entero más próxima por encima,
<code>Math.round(6.4);</code>	// Redondea a la parte entera más próxima,
<code>Math.abs(x);</code>	// Devuelve el valor absoluto de un número

- c) Clase String: Es similar a un array, con índices que van desde el 0 para el primer carácter hasta el último.

Propiedades y Metodos	Valor devuelto
<code>"Aplicaciones"[4]</code>	// Devuelve "c"
<code>Aplicaciones.length</code>	// Devuelve la longitud de un string, es propiedad y debe ir sin paréntesis de función.
<code>"Aplicaciones".charCodeAt(4);</code>	// Devuelve el carácter 4 en formato UNICODE es decir de "c", el 99.
<code>"Aplicaciones".indexOf("cion");</code>	// Devuelve el índice donde comienza el string "cion", el 6
<code>"Aplicaciones".substring(4,8);</code>	// Devuelve parte del string desde la posición 4 hasta la posición (8-1) es decir "caci"
<code>"lun mar mie jue vie".split(" ");</code>	Convierte un string en array. Cada palabra debe estar separado por un semarador en común como un espacio. <code>var miArray = "lun mar mie jue vie".split(" ");</code> <code>console.log(miArray);</code> //Devuelve ["lun", "mar", "mie", "jue", "vie"]

- d) Array: En una colección de datos que pueden contener números, strings, objetos, etc.

```
var miArray = [];
```

o

```
var miArray = new Array();
```

Ejemplos:

```
var miArray = [1, 2, 3, 4]; // Array de números
var miArray = ["Hola", "que", "tal"]; // Array de Strings
var miArray = [ {propiedad: "valor1" }, { propiedad: "valor2" }];

// Array de objetos
var miArray = [[2, 4], [3, 6]]; // Array de arrays, (Matriz);
var miArray = [1, true, [3,2], "Hola", {clave: "valor"}]; // Array mixto
```

e) Desestructuración de Arrays

```
let a, b, rest;
[a, b] = [10, 20];

console.log(a);
// expected output: 10

console.log(b);
// expected output: 20

[a, b, ...rest] = [10, 20, 30, 40, 50];

console.log(rest);
// expected output: Array [30,40,50]
```

- Objetos y Clases en JS

a) Objetos: Son simples colecciones de pares nombre-valor.

```
var obj = new Object({ propiedad: "valor" });

y:

var obj = { propiedad: "valor" };
```

```
function Persona(nombre, edad) {
  this.nombre = nombre;
  this.edad = edad;
}

// Definir un objeto
var Tu = new Persona("Tu", 24);
// Estamos creando una nueva persona llamada "Tu"
// (que fue el primer parametro, y su edad, el segundo)
```

- Clases en ECMAScript6: ES6 aporta un azúcar sintáctico para declarar una clase como en la mayoría de los lenguajes de programación orientados a objetos, pero por debajo sigue siendo una función prototipal.

```

class Inventario {
  constructor(nombre) {
    this.nombre = nombre;
    this.articulos = [];
  }
  add (nombre, cantidad) {
    this.articulos[nombre] = cantidad;
  }
  borrar (nombre) {
    delete this.articulos[nombre]
  }
  cantidad (nombre) {
    return this.articulos[nombre]
  }
  getNombre () {
    return this.nombre;
  }
}

```

```

class Vehiculo {
  constructor (tipo, nombre, ruedas) {
    this.tipo = tipo;
    this.nombre = nombre;
    this.ruedas = ruedas
  }
  getRuedas () {
    return this.ruedas
  }
  arrancar () {
    console.log(`Arrancando el ${this.nombre}`)
  }
  aparcas () {
    console.log(`Aparcando el ${this.nombre}`)
  }
}

```

- Colecciones con Llave: Los objetos Map y Set contienen elementos cuya iteración se da conforme al orden de inserción.
 - a) El Objeto Map: Es un simple mapeo clave/valor y puede iterar sobre sus elementos en el orden de inserción.

El siguiente código muestra algunas operaciones básicas con un Map.

```

var sonidos = new Map();
sonidos.set("perro", "guau");
sonidos.set("gato", "miau");
sonidos.set("oveja", "beee");
sonidos.size; // 3
sonidos.get("zorro"); // undefined
sonidos.has("pájaro"); // false
sonidos.delete("perro");

for (var [clave, valor] of sonidos) {
  console.log(clave + " hace " + valor);
}
// "gato hace miau"
// "oveja hace beee"

```

- b) Comparación de Object y Map: Usa mapas preferiblemente cuando las claves son desconocidas hasta el tiempo de ejecución,

o cuando todas las claves son del mismo tipo y todos los valores son del mismo tipo.

- ¿Qué es HTML Canvas?: Es solo un contenedor de gráficos. Debes utilizar un script para dibujar los gráficos.

Etiqueta Canvas

```
<canvas id="myCanvas" width="200" height="100"></canvas>
```

Dibujar en el lienzo con JavaScript

```
<script>
const canvas = document.getElementById("myCanvas");
const ctx = canvas.getContext("2d");

ctx.fillStyle = "#FF0000";
ctx.fillRect(0, 0, 150, 75);
</script>
```

PRACTICA CALIFICADA

Código:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Obtener la hora actual</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <button id="btnMostrarHora">Mostrar hora actual</button>
    <p id="horaActual"></p>

    <script src="script.js"></script>
    <hr />
    <section>
      <h3>
        EJERCICIO 1 - solucion del profesor Manipular el html con un
        click
      </h3>
      <p id="parrafo1">Texto1</p>
      <button type="button" onclick="modificandoDom()" class="btn
        succes">
        hacer click
      </button>
    </script>
    function modificandoDom() {
```

```

        let parrafo1 = document.getElementById("parrafo1");
        parrafo1.innerHTML = "cambiando el DOM" + Date();
    }
</script>
</section>
<hr />
<section>
    <h3>Ejercicio 2- Manipular html con un AddEventListener</h3>
    <form action="">
        <label for="">Ingrese texto:</label>
        <input type="text" name="" id="idinput" />
        <p id="idparrafo">texto2</p>
    </form>

    <script>
        let txt = document.getElementById("idinput");
        txt.addEventListener("keyup", cambiandoParrafo);
        function cambiandoParrafo() {
            let parrafo1 = document.getElementById("idparrafo");
            parrafo1.innerHTML = txt.value;
        }
    </script>
</section>
<hr />
<section>
    <form id="formularioSuma">
        <label for="num1">Número 1:</label>
        <input type="number" id="num1" name="num1" required />
        <label for="num2">Número 2:</label>
        <input type="number" id="num2" name="num2" required />
        <button type="button" id="btnSumar">Sumar</button>
        <p id="resultadoSuma"></p>
    </form>

    <script src="script.js"></script>
</section>
<hr />
<section>
    <h3>Calculadora de Suma</h3>

    <form id="formularioSuma">
        <label for="num1">Número 1:</label>
        <input type="number" id="num1" name="num1" required />
        <label for="num2">Número 2:</label>
        <input type="number" id="num2" name="num2" required />
        <button type="button" id="btnSumar">Sumar</button>
        <p id="resultadoSuma"></p>
    </form>

```



```

        <p id="resultadoSuma"></p>

        <script src="script.js"></script>
    </section>
    <hr />
    <section>
        <h1>Tabla de estudiantes</h1>
        <table id="tablaEstudiantes">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Apellidos</th>
                    <th>Nombres</th>
                    <th>Dirección</th>
                    <th>Teléfono</th>
                </tr>
            </thead>
            <tbody></tbody>
        </table>

        <script src="script.js"></script>
    </section>

    <hr />
</body>
</html>

<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>Obtener la hora actual</title>
        <link rel="stylesheet" href="style.css" />
    </head>
    <body>
        <button id="btnMostrarHora">Mostrar hora actual</button>
        <p id="horaActual"></p>

        <script src="script.js"></script>
        <hr />
        <section>
            <h3>
                EJERCICIO 1 - solucion del profesor Manipular el html con un
click
            </h3>
            <p id="parrafo1">Texto1</p>

```

```

    <button type="button" onclick="modificandoDom()" class="btn
succes">
        hacer click
    </button>
    <script>
        function modificandoDom() {
            let parrafo1 = document.getElementById("parrafo1");
            parrafo1.innerHTML = "cambiando el DOM" + Date();
        }
    </script>
</section>
<hr />
<section>
    <h3>Ejercicio 2- Manipular html con un AddEventListener</h3>
    <form action="">
        <label for="">Ingrese texto:</label>
        <input type="text" name="" id="idinput" />
        <p id="idparrafo">texto2</p>
    </form>

    <script>
        let txt = document.getElementById("idinput");
        txt.addEventListener("keyup", cambiandoParrafo);
        function cambiandoParrafo() {
            let parrafo1 = document.getElementById("idparrafo");
            parrafo1.innerHTML = txt.value;
        }
    </script>
</section>
<hr />
<section>
    <form id="formularioSuma">
        <label for="num1">Número 1:</label>
        <input type="number" id="num1" name="num1" required />
        <label for="num2">Número 2:</label>
        <input type="number" id="num2" name="num2" required />
        <button type="button" id="btnSumar">Sumar</button>
        <p id="resultadoSuma"></p>
    </form>

    <script src="script.js"></script>
</section>
<hr />
<section>
    <h3>Calculadora de Suma</h3>

    <form id="formularioSuma">
        <label for="num1">Número 1:</label>
        <input type="number" id="num1" name="num1" required />

```

```

    <label for="num2">Número 2:</label>
    <input type="number" id="num2" name="num2" required />
    <button type="button" id="btnSumar">Sumar</button>
    <p id="resultadoSuma"></p>
  </form>

  <p id="resultadoSuma"></p>

  <script src="script.js"></script>
</section>
<hr />
<section>
  <h1>Tabla de estudiantes</h1>
  <table id="tablaEstudiantes">
    <thead>
      <tr>
        <th>ID</th>
        <th>Apellidos</th>
        <th>Nombres</th>
        <th>Dirección</th>
        <th>Teléfono</th>
      </tr>
    </thead>
    <tbody></tbody>
  </table>

  <script src="script.js"></script>
</section>

  <hr />
</body>
</html>

const estudiantes = [
  { id: 1, nombre: "Juan", direccion: "Calle A #123", telefono: "555-1234" },
  { id: 2, nombre: "María", direccion: "Avenida B #456", telefono: "555-5678" },
  { id: 3, nombre: "Carlos", direccion: "Calle C #789", telefono: "555-9012" },
  { id: 4, nombre: "Laura", direccion: "Avenida D #321", telefono: "555-3456" },
  { id: 5, nombre: "Pedro", direccion: "Calle E #654", telefono: "555-7890" },
  { id: 6, nombre: "Ana", direccion: "Avenida F #987", telefono: "555-2345" },
  { id: 7, nombre: "Pablo", direccion: "Calle G #210", telefono: "555-6789" },
];

```

```

    { id: 8, nombre: "Sofía", direccion: "Avenida H #543", telefono: "555-1234" },
    { id: 9, nombre: "Luis", direccion: "Calle I #876", telefono: "555-5678" },
    {
      id: 10,
      nombre: "Elena",
      direccion: "Avenida J #109",
      telefono: "555-9012",
    },
  ],
];

```

```

document.addEventListener("DOMContentLoaded", () => {
  const tablaBody = document.querySelector("#tablaEstudiantes tbody");

  // Recorrer el array de estudiantes y crear filas de tabla
  estudiantes.forEach((estudiante) => {
    const fila = document.createElement("tr");
    fila.innerHTML = `
      <td>${estudiante.id}</td>
      <td>${estudiante.nombre}</td>
      <td>${estudiante.direccion}</td>
      <td>${estudiante.telefono}</td>
    `;
    tablaBody.appendChild(fila);
  });
});

```

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Tabla de Estudiantes</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Tabla de Estudiantes</h1>

    <!-- Botón para mostrar la tabla -->

    <table id="tablaEstudiantes">
      <thead>
        <tr>
          <th>ID</th>
          <th>Nombre</th>

```

```

        <th>Dirección</th>
        <th>Teléfono</th>
    </tr>
</thead>
<tbody>
    <!-- Aquí se agregarán dinámicamente las filas de la tabla -->
</tbody>
</table>

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

<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0"
    />
        <title>Tabla de Estudiantes</title>
        <link rel="stylesheet" href="styles.css" />
    </head>
    <body>
        <script src="script.js"></script>
    </body>
</html>

class Estudiante {
    constructor(id, apellidos, nombres, direccion, telefono) {
        this.id = id;
        this.apellidos = apellidos;
        this.nombres = nombres;
        this.direccion = direccion;
        this.telefono = telefono;
    }
}

const estudiantes = [
    new Estudiante(1, "González", "Juan", "Calle 123", "555-1234"),
    new Estudiante(2, "Pérez", "María", "Avenida 456", "555-5678"),
    new Estudiante(3, "Rodríguez", "Carlos", "Plaza Principal", "555-
9876"),
    new Estudiante(4, "Martínez", "Laura", "Ruta 789", "555-4321"),
    new Estudiante(5, "Gómez", "Pedro", "Boulevard 321", "555-8765"),
    new Estudiante(6, "Díaz", "Ana", "Callejon 654", "555-2345"),
    new Estudiante(7, "Sánchez", "Jorge", "Pasaje 987", "555-7654"),
    new Estudiante(8, "López", "Sofía", "Avenida Sur 456", "555-3456"),
    new Estudiante(9, "Ramírez", "Diego", "Esquina 789", "555-6543"),

```

```

    new Estudiante(10, "Fernández", "Elena", "Calle Norte 321", "555-8765"),
  ];

document.addEventListener("DOMContentLoaded", () => {
  const tabla = document.createElement("table");
  tabla.innerHTML = `
    <thead>
      <tr>
        <th>ID</th>
        <th>Apellidos</th>
        <th>Nombres</th>
        <th>Dirección</th>
        <th>Teléfono</th>
        <th>Acciones</th>
      </tr>
    </thead>
    <tbody>
      ${estudiantes
        .map(
          (estudiante) => `
            <tr>
              <td>${estudiante.id}</td>
              <td>${estudiante.apellidos}</td>
              <td class="nombre">${estudiante.nombres}</td>
              <td>${estudiante.direccion}</td>
              <td>${estudiante.telefono}</td>
              <td>
                <button class="editar-btn">Editar</button>
                <button class="guardar-btn" style="display:
none;">Guardar</button>
              </td>
            </tr>
          `
        )
        .join("")}
    </tbody>
  `;

  document.body.appendChild(tabla);

  // Agregar event listeners para botones de editar y guardar
  const editarBtns = document.querySelectorAll(".editar-btn");
  const guardarBtns = document.querySelectorAll(".guardar-btn");

  editarBtns.forEach((btn, index) => {
    btn.addEventListener("click", () => {
      const nombreCell = document.querySelectorAll(".nombre")[index];
      const nombreOriginal = nombreCell.textContent;

```

```

    const input = document.createElement("input");
    input.type = "text";
    input.value = nombreOriginal;
    nombreCell.textContent = "";
    nombreCell.appendChild(input);

    btn.style.display = "none";
    guardarBtns[index].style.display = "inline-block";
  });
});

guardarBtns.forEach((btn, index) => {
  btn.addEventListener("click", () => {
    const nombreCell = document.querySelectorAll(".nombre")[index];
    const nuevoNombre = nombreCell.querySelector("input").value;
    nombreCell.textContent = nuevoNombre;

    btn.style.display = "none";
    editarBtns[index].style.display = "inline-block";
  });
});
});

```

Vista del Navegador:

Mostrar hora actual

La hora actual es: 22:41:49

EJERCICIO 1 - solucion del profesor Manipular el html con un click

Texto1

hacer click

Ejercicio 2- Manipular html con un AddEventListener

Ingrese texto:

texto2

Número 1: Número 2:

Calculadora de Suma

Número 1: Número 2:

Tabla de
Estudiantes

ID	Nombre	Dirección	Teléfono
1	Juan	Calle A #123	555-1234
2	Maria	Avenida B #456	555-5678
3	Carlos	Calle C #789	555-9012
4	Laura	Avenida D #321	555-3456
5	Pedro	Calle E #654	555-7890
6	Ana	Avenida F #987	555-2345
7	Pablo	Calle G #210	555-6789
8	Sofia	Avenida H #543	555-1234
9	Luis	Calle I #876	555-5678
10	Elena	Avenida J #109	555-9012

ID	Apellidos	Nombres	Dirección	Teléfono	Acciones
1	González	Juan	Calle 123	555-1234	<button>Editar</button>
2	Pérez	María	Avenida 456	555-5678	<button>Editar</button>
3	Rodríguez	Carlos	Plaza Principal	555-9876	<button>Editar</button>
4	Martínez	Laura	Ruta 789	555-4321	<button>Editar</button>
5	Gómez	Pedro	Boulevard 321	555-8765	<button>Editar</button>
6	Díaz	Ana	Callejon 654	555-2345	<button>Editar</button>
7	Sánchez	Jorge	Pasaje 987	555-7654	<button>Editar</button>
8	López	Sofía	Avenida Sur 456	555-3456	<button>Editar</button>
9	Ramírez	Diego	Esquina 789	555-6543	<button>Editar</button>
10	Fernández	Elena	Calle Norte 321	555-8765	<button>Editar</button>

ID	Apellidos	Nombres	Dirección	Teléfono	Acciones
1	González	<input type="text" value="Don Mario"/>	Calle 123	555-1234	<button>Guardar</button>
2	Pérez	María	Avenida 456	555-5678	<button>Editar</button>
3	Rodríguez	Carlos	Plaza Principal	555-9876	<button>Editar</button>
4	Martínez	Laura	Ruta 789	555-4321	<button>Editar</button>

ID	Apellidos	Nombres	Dirección	Teléfono	Acciones
1	González	Don Mario	Calle 123	555-1234	<button>Editar</button>
2	Pérez	María	Avenida 456	555-5678	<button>Editar</button>
3	Rodríguez	Carlos	Plaza Principal	555-9876	<button>Editar</button>