Programación Orientada a Objetos

Clase 8

KOICA

HANDONG GLOBAL UNIVERSITY

UNA

OBJETIVOS DE LA CLASE 8

- **Manejar los conceptos de programación orientada a objetos, la clase Date, la clase Array, la clase Math y la clase String.**
- **Implementar los códigos de ejemplos propuestos en clase.**

Programación orientada a objetos

- Un objeto es una estructura que contiene tanto las variables (llamadas propiedades) como las funciones que manipulan dichas variables (llamadas métodos). A partir de esta estructura se ha creado un nuevo modelo de programación (la programación orientada a objetos) que atribuye a los mismos propiedades como herencia o polimorfismo.
- Como veremos, JavaScript simplifica en algo este modelo y hace una programación híbrida entre la programación estructurada y la programación orientada a objetos.

Programación orientada a objetos

- El modelo de la programación orientada a objetos normal y corriente separa los mismos en dos: clases e instancias (objetos). Las primeras son entes más abstractos que definen un conjunto determinado de objetos. Las segundas son miembros de una clase, poseyendo las mismas propiedades que la clase a la cual pertenecen.

Propiedades y métodos

- Para acceder a los métodos y propiedades de un objeto debemos utilizar la siguiente sintaxis:

  objeto.propiedad

  objeto.metodo(parametros)

Conceptos Básicos

- Objetos: Son todas las cosas con identidad propia. Se relacionan entre si. Poseen características (atributos) y tienen responsabilidades (funciones, métodos) que deben cumplir. Son ejemplares (instancias) de una clase y conocen a la clase a la cual pertenecen.
- Atributos o propiedades: Son las características, cualidades distintivas de cada objeto. Deben ser mínimos para poder realizar todas las operaciones que requiere la aplicación.

Ejemplos de objetos del mundo real:

- Casa:
  atributos: tamaño, precio, cantidad de habitaciones, etc.;
  responsabilidades: comodidad, seguridad, etc.
- Mesa:
  atributos: altura, largo, ancho, etc.;
  responsabilidades: contener elementos.
- Ventana:
  atributos: tamaño, color, etc.;
  responsabilidades: abrirse, cerrarse, etc.

Ejemplos de objetos del mundo de la programación:

- Ventana:
    - atributos: tamaño, color, etc.;
    - responsabilidades: mostrar título,achicarse, etc.

Responsabilidades o Métodos

- Son las responsabilidades que debe cumplir la clase.
- El objetivo de un método es ejecutar las actividades que tiene encomendada la clase.
- Es un algoritmo (conjunto de operaciones) que se ejecuta en respuesta a un mensaje; respuestas a mensajes para satisfacer peticiones.
- Un método consiste en el nombre de la operación y sus argumentos. El nombre del método identifica una operación que se ejecuta.

Responsabilidades o Métodos

- Un método está determinado por la clase del objeto receptor, todos los objetos de una clase usan el mismo método en respuesta a mensajes similares.
- La interpretación de un mensaje (selección del método ejecutado) depende del receptor y puede variar con distintos receptores, es decir, puede variar de una clase a otra.

Clases

- Una clase es un molde para objetos que poseen las mismas características (que pueden recibir los mismos mensajes y responden de la misma manera).
- Una clase es una representación de una idea o concepto. Unidad que encapsula códigos y datos para los métodos (operaciones).
- Todos los ejemplares de una clase se comportan de forma similar (invocan el mismo método) en respuesta a mensajes similares.

Clases

- La clase a la cual pertenece un objeto determina el comportamiento del objeto.
- Una clase tiene encomendadas actividades que ejecutan los métodos.
- Las clases están definidas por:

  - Atributos (Propiedades),

  - Comportamiento (operaciones o métodos) y

  - Relaciones con otros objetos.

- Una aplicación es un conjunto de objetos de determinadas clases.

1. CLASE DATE

Clase Date

- JavaScript dispone de varias clases predefinidas para acceder a muchas de las funciones normales de cualquier lenguaje, como puede ser el manejo de vectores o el de fechas.
- Esta clase nos permitirá manejar fechas y horas. Se invoca así:

```
fecha = new Date();//creación de un objeto de la clase Date

fecha = new Date(año, mes, dia);

fecha = new Date(año, mes, dia, hora, minuto, segundo);
```

Clase Date

- Si no utilizamos parámetros, el objeto fecha contendrá la fecha y hora actual, recuperadas del reloj de nuestra computadora. En caso contrario hay que tener en cuenta que los meses comienzan por cero.
- Así, por ejemplo:

```
navidad2020 = new Date(2020, 11, 25)
```

El objeto Date dispone, entre otros, de los siguientes métodos:

- getFullYear()
- setFullYear(año)
- getMonth()
- setMonth(mes)
- getDate()
- setDate(dia)
- getHours()
- setHours(horas)
- getMinutes()
- setMinutes(minutos)
- getSeconds()
- setSeconds(segundos)

Obtienen y colocan, respectivamente, el mes, día, hora, minuto y segundo de la fecha.

El objeto Date dispone, entre otros, de los siguientes métodos:

getDay()

Devuelve el día de la semana de la fecha en forma de número que va del 0 (domingo) al 6 (sábado)

Ejemplo

Mostrar en una página la fecha y la hora actual.

```
<script>
        function mostrarFechaHora() {
            let fecha;
            fecha = new Date();
            document.write('Hoy es ');
            document.write(fecha.getDate() + '/');
            document.write((fecha.getMonth() + 1) + '/');
            document.write(fecha.getFullYear());
            document.write('<br>');
            document.write('Es la hora ');
            document.write(fecha.getHours() + ':');
            document.write(fecha.getMinutes() + ':');
            document.write(fecha.getSeconds());
        }

        //Llamada a la función
        mostrarFechaHora();
</script>
```

Clase Date

- En este problema hemos creado un objeto de la clase Date, para ello utilizamos la siguiente sintaxis:

```
let fecha;

fecha = new Date();
```

- Luego llamamos una serie de métodos que nos retornan datos sobre la fecha y hora actual del equipo de computación donde se está ejecutando el navegador.

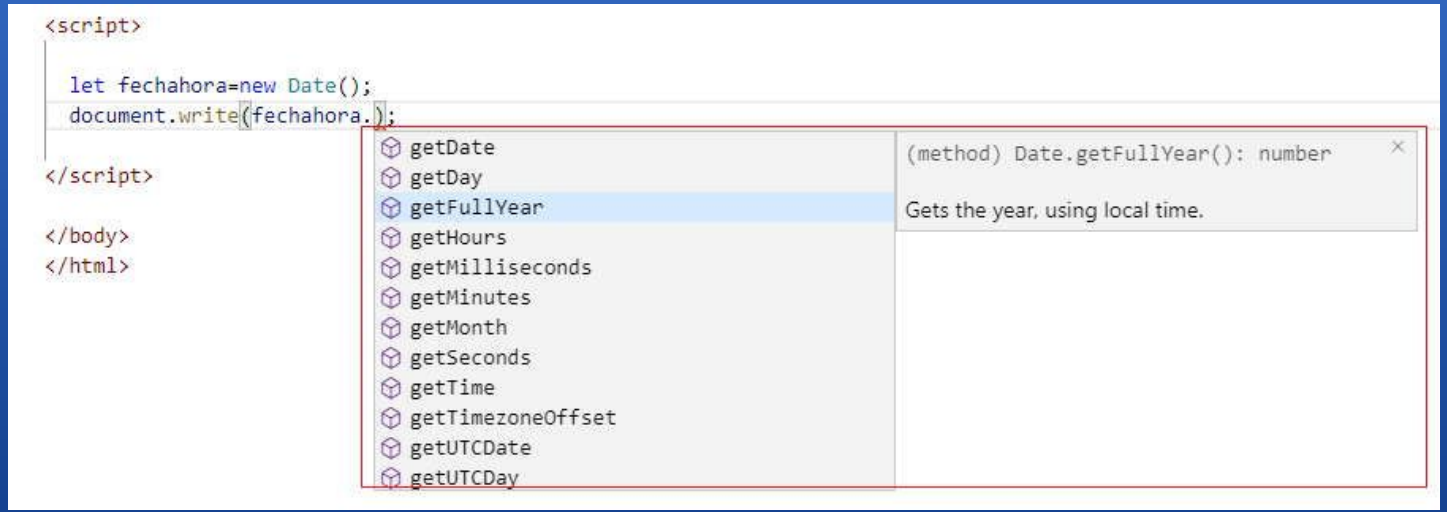
Clase Date

- Es bueno notar que para llamar a los métodos disponemos:

  <nombre de objeto>.<nombre de método>(parámetros)

Acotaciones

Muchos editores de texto como el VSCode nos facilitan recordar los nombres de métodos cuando escribimos el nombre del objeto y el carácter punto:



Problema

Confeccionar un programa que muestre en qué cuatrimestre del año nos encontramos. Para esto obtener el mes.

```
<script>
        let fecha
        fecha = new Date();
        let mes = fecha.getMonth();
        if (mes < 4) {
            document.write('Nos encontramos en el primer
cuatrimestre del año');
        } else {
            if (mes < 8) {
                document.write('Nos encontramos en el
segundo cuatrimestre del año');
            } else {
                document.write('Nos encontramos en el
tercer cuatrimestre del año');
            }
        }
    </script>
```

Problema

Confeccionar una función que nos retorne un string con el siguiente formato:

Hoy es Lunes 9 de Agosto de 2021

Para poder recuperar el día de la semana debemos llamar al método:

```
let diaSemana=fecha.getDay();
```

El método getDay() devuelve el día de la semana de la fecha especificada, siendo 0 (Domingo) el primer día.

```
<script>
    function retornarDiaSemana(dia) {
        switch (dia) {
            case 0:
                return 'Domingo';
            case 1:
                return 'Lunes';
            case 2:
                return 'Martes';
            case 3:
                return 'Miércoles';
            case 4:
                return 'Jueves';
            case 5:
                return 'Viernes';
            case 6:
                return 'Sábado';
            default:
                return 'día inválido';
        }
    }
```

```
    function retornarMes(mes) {
        switch (mes) {
            case 0:
                return 'Enero';
            case 1:
                return 'Febrero';
            case 2:
                return 'Marzo';
            case 3:
                return 'Abril';
            case 4:
                return 'Mayo';
            case 5:
                return 'Junio';
            case 6:
                return 'Julio';
            case 7:
                return 'Agosto';
            case 8:
                return 'Septiembre';
            case 9:
                return 'Octubre';
            case 10:
                return 'Noviembre';
            case 11:
                return 'Diciembre';
            default:
                return 'mes inválido';
        }
    }
```

```
    function retornarFechaTexto() {
        let fecha = new Date();
        let cadena = 'Hoy es ' + retornarDiaSemana(fecha.getDay()) + ' ' + fecha.getDate() + ' de ' + retornarMes(fecha.getMonth()) + ' de ' + fecha.getFullYear();
        return cadena;
    }

document.write(retornarFechaTexto());
</script>
```

2. CLASE ARRAY

Clase Array

- Un vector es una estructura de datos que permite almacenar un CONJUNTO de datos.
- Con un único nombre se define un vector y por medio de un subíndice hacemos referencia a cada elemento del mismo (componente).

Ejemplo 1:

Crear un vector para almacenar los cinco sueldos de operarios y luego mostrar el total de gastos en sueldos (cada actividad en una función).

```
<script>
    function cargar(sueldos) {
        for (let f = 0; f < sueldos.length; f++) {
            let v = prompt('Ingrese sueldo:');
            sueldos[f] = parseInt(v);
        }
    }

    function calcularGastos(sueldos) {
        let total = 0;
        for (let f = 0; f < sueldos.length; f++) {
            total = total + sueldos[f];
        }
        document.write('Listado de sueldos<br>');
        for (let f = 0; f < sueldos.length; f++) {
            document.write(sueldos[f] + '<br>');
        }
        document.write('Total de gastos en sueldos:' + total);
    }

    let sueldos;
    sueldos = new Array(5);
    cargar(sueldos);
    calcularGastos(sueldos);
</script>
```

Recordemos que

- el programa comienza a ejecutarse a partir de las líneas que se encuentran fuera de las funciones:

```
let sueldos;

sueldos = new Array(5);

cargar(sueldos);

calcularGastos(sueldos);
```

Lo primero

- Definimos una variable y posteriormente creamos un objeto de la clase Array, indicándole que queremos almacenar 5 valores.
- Llamamos a la función cargar enviándole el vector. En la función, a través de un ciclo for recorremos las distintas componentes del vector y almacenamos valores enteros que ingresamos por teclado.
- Para conocer el tamaño del vector accedemos a la propiedad length de la clase Array.
- Cuando en la función 'cargar' ingresamos los elementos del arreglo, sucede que se modifica la variable 'sueldos' definida en la parte inferior.
- En la segunda función sumamos todas las componentes del vector, imprimimos en la página los valores y el total de gastos.

Ejemplo 2:

Crear un vector con elementos de tipo string. Almacenar los meses del año. En una función solicitar el ingreso de un número entre 1 y 12. Mostrar a qué mes corresponde y cuántos días tiene dicho mes.

```
<script>
    function mostrarFecha(meses, dias) {
        let num = parseInt(prompt('Ingrese número de mes:'));
        document.write('Corresponde al mes:' + meses[num - 1]);
        document.write('<br>');
        document.write('Tiene ' + dias[num - 1] + ' días');
    }

    let meses;
    meses = new Array(12);
    meses[0] = 'Enero';
    meses[1] = 'Febrero';
    meses[2] = 'Marzo';
    meses[3] = 'Abril';
    meses[4] = 'Mayo';
    meses[5] = 'Junio';
    meses[6] = 'Julio';
    meses[7] = 'Agosto';
    meses[8] = 'Septiembre';
    meses[9] = 'Octubre';
    meses[10] = 'Noviembre';
    meses[11] = 'Diciembre';
    let dias;
    dias = new Array(12);
    dias[0] = 31;
    dias[1] = 28;
    dias[2] = 31;
    dias[3] = 30;
    dias[4] = 31;
    dias[5] = 30;
    dias[6] = 31;
    dias[7] = 31;
    dias[8] = 30;
    dias[9] = 31;
    dias[10] = 30;
    dias[11] = 31;
    mostrarFecha(meses, dias);
</script>
```

En este problema

- Definimos dos vectores, uno para almacenar los meses y otro los días. Decimos que se trata de vectores paralelos porque en la componente cero del vector meses almacenamos el string 'Enero' y en el vector dias, la cantidad de días del mes de enero.
- Es importante notar que cuando imprimimos, disponemos como subíndice el valor ingresado menos 1, esto debido a que normalmente el operador de nuestro programa carga un valor comprendido entre 1 y 12. Recordar que los vectores comienzan a numerarse a partir de la componente cero.

```
document.write('Corresponde al mes:'+meses[num-1]);
```

Acotaciones

Es común utilizar una sintaxis más concisa para inicializar los dos vectores:

```
<script>
        function mostrarFecha(meses, dias) {
            let num = parseInt(prompt('Ingrese número de mes:'));
            document.write('Corresponde al mes:' + meses[num - 1]);
            document.write('<br>');
            document.write('Tiene ' + dias[num - 1] + ' días');
        }

        let meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio', 'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre'];
        let dias = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31];
        mostrarFecha(meses, dias);
    </script>
```

Hay mucho por ver de la clase Array más adelante, por ejemplo:

- Podemos definir arreglos con elementos de distinto tipo en cada componente:

```
let persona = ['Diego Martinez', 50, 78.50];

document.write(persona[0] + ' tiene una edad de ' + persona[1] + ' y un peso de ' + persona[2])
```

Por pantalla aparece:

Diego Martinez tiene una edad de 50 y un peso de 78.5

Hay mucho por ver de la clase Array más adelante, por ejemplo:

- Podemos definir un vector sin indicar el tamaño del mismo y luego cargar en cualquier posición:

```
let ingresos = [];

ingresos[3] = 1000;

ingresos[5] = 5000;

for (let f = 0; f <= 6; f++) {

    document.write(ingresos[f] + '<br>')

}
```

Hay mucho por ver de la clase Array más adelante, por ejemplo:

- Las componentes que no han sido inicializadas retornan el valor undefined.

undefined

undefined

undefined

1000

undefined

5000

undefined

Hay mucho por ver de la clase Array más adelante, por ejemplo:

- Podemos definir componentes del arreglo que sean otro arreglo (crear estructuras de datos tan complejas como necesitemos):

```
let matrizIdentidad = [
        [1, 0, 0],
        [0, 1, 0],
        [0, 0, 1]
    ];
```

```
for (let f = 0; f < 3; f++) {
        for (let c = 0; c < 3; c++) {
                document.write(matrizIdentidad[f][c] + ' ');
        }
        document.write('<br>');
}
```

Hay mucho por ver de la clase Array más adelante, por ejemplo:

- Luego tenemos como resultado en la pantalla:

1 0 0

0 1 0

0 0 1

Luego veremos que tenemos otras muchísimas funcionalidades con esta estructura de datos.

Problema

Desarrollar un programa que permita ingresar un vector de 8 elementos, e informe:

- El valor acumulado de todos los elementos del vector.
- El valor acumulado de los elementos del vector que sean mayores a 36.
- Cantidad de valores mayores a 50.

```
<script>
        function cargar(vec) {
            for (let f = 0; f < vec.length; f++) {
                let valor = prompt('Ingrese componente:');
                vec[f] = parseInt(valor);
            }
        }

        function sumar(vec) {
            let suma = 0;
            for (let f = 0; f < vec.length; f++) {
                suma = suma + vec[f];
            }
            document.write('Valor acumulado de las componentes:' + suma + '<br>');
        }
```

```
 function sumarMayor36(vec) {
            let suma = 0;
            for (let f = 0; f < vec.length; f++) {
                if (vec[f] > 36) {
                    suma = suma + vec[f];
                }
            }
            document.write('Valor acumulado de las componentes mayores a 36:' + suma + '<br>');
        }
    function cantidadMayores50(vec) {
            let cant = 0;
            for (let f = 0; f < vec.length; f++) {
                if (vec[f] > 50) {
                    cant = cant + 1;
                }
            }
            document.write('Cantidad de componentes mayores a 50:' + cant + '<br>');
        }

    let vec = new Array(8);
        cargar(vec);
        sumar(vec);
        sumarMayor36(vec);
        cantidadMayores50(vec);
    </script>
```

Problema

Realizar un programa que pida la carga de dos vectores numéricos. Obtener la suma de los dos vectores, dicho resultado guardarlo en un tercer vector del mismo tamaño. Sumar componente a componente. El tamaño del vector es a elección.

```
<script>
        function cargarVectores(vec1, vec2) {

            for (let f = 0; f < vec1.length; f++) {
                let valor = prompt('Ingrese componente del primer vector:', '');
                vec1[f] = parseInt(valor);
            }
            for (let f = 0; f < vec2.length; f++) {
                let valor = prompt('Ingrese componente del segundo vector:', '');
                vec2[f] = parseInt(valor);
            }
        }
```

```
function sumarVectores(vec1, vec2, vecsuma) {
            for (let f = 0; f < vecsuma.length; f++) {
                vecsuma[f] = vec1[f] + vec2[f];
            }
        }

        function imprimirVector(vecsuma) {
            for (let f = 0; f < vecsuma.length; f++) {
                document.write(vecsuma[f] + ' ');
            }
        }

        let vec1 = new Array(5);
        let vec2 = new Array(5);
        let vecsuma = new Array(5);
        cargarVectores(vec1, vec2);
        sumarVectores(vec1, vec2, vecsuma);
        imprimirVector(vecsuma);
    </script>
```

3. CLASE MATH

Clase Math

Esta clase es un contenedor que tiene diversas constantes (como Math.E y Math.PI) y un conjunto de métodos matemáticos:

Método	Descripción	Expresión de ejemplo	Resultado
abs	Valor absoluto	Math.abs(-2)	2
sin, cos, tan	Funciones trigonométricas, reciben el argumento en radianes	Math.cos(Math.PI)	-1
asin, acos, atan	Funciones trigonométricas inversas	Math.asin(1)	1.57

Clase Math

Esta clase es un contenedor que tiene diversas constantes (como Math.E y Math.PI) y un conjunto de métodos matemáticos:

Método	Descripción	Expresión de ejemplo	Resultado
exp, log	Exponenciación y logaritmo, base E	Math.log(Math.E)	1
ceil	Devuelve el entero más pequeño mayor o igual al argumento	Math.ceil(-2.7)	-2
floor	Devuelve el entero más grande menor o igual al argumento	Math.floor(-2.7)	-3
round	Devuelve el entero más cercano o igual al argumento	Math.round(-2.7)	-3

Clase Math

Esta clase es un contenedor que tiene diversas constantes (como Math.E y Math.PI) y un conjunto de métodos matemáticos:

Método	Descripción	Expresión de ejemplo	Resultado
min, max	Devuelve el menor (o mayor) de sus dos argumentos	Math.min(2,4)	2
pow	Exponenciación, siendo el primer argumento la base y el segundo el exponente	Math.pow(2,3)	8
sqrt	Raíz cuadrada	Math.sqrt(25)	5
random	Genera un valor aleatorio comprendido entre 0 y 1.	Math.random()	Math.random()

Ejemplo

Confeccionar un programa que permita cargar un valor comprendido entre 1 y 10. Luego generar un valor aleatorio entre 1 y 10, mostrar un mensaje con el número sorteado e indicar si ganó o perdió:

```
<script>
        let selec = parseInt(prompt('Ingrese un valor entre 1 y 10'));
        let num = parseInt(Math.random() * 10) + 1;
        if (num == selec)
            document.write('Ganó, el número que se sorteó es el ' + num);
        else
            document.write('Lo siento, se sorteó el valor ' + num + ' y usted eligió el ' + selec);
    </script>
```

Clase Math

Para generar un valor aleatorio comprendido entre 1 y 10 debemos plantear lo siguiente:

```
let num = parseInt(Math.random() * 10) + 1;
```

Al multiplicar Math.random() por 10, nos genera un valor aleatorio comprendido entre un valor mayor a 0 y menor a 10, luego, con la función parseInt, obtenemos sólo la parte entera. Finalmente sumamos uno.

Clase Math

El valor que cargó el operador se encuentra en:

```
let selec = parseInt(prompt('Ingrese un valor entre 1 y 10'));
```

Con un simple if validamos si coinciden los valores (el generado y el ingresado por teclado)

4. CLASE STRING

Clase String

- Un string consiste en uno o más caracteres encerrados entre simple o doble comillas.

Concatenación de cadenas (+)

- JavaScript permite concatenar cadenas utilizando el operador +.
- El siguiente fragmento de código concatena tres cadenas para producir su salida:

```
let final='La entrada tiene ' + contador + ' caracteres.';
```

- Dos de las cadenas concatenadas son cadenas literales. La del medio es un entero que automáticamente se convierte a cadena y luego se concatena con las otras.

Propiedad length

- Retorna la cantidad de caracteres de un objeto String.

```
let nom='Juan';
```

- document.write(nom.length); //Resultado 4

Métodos

- charAt(pos)
- Retorna el carácter del índice especificado. Comienzan a numerarse de la posición cero.

```
let nombre = 'juan';

let caracterPrimero = nombre.charAt(0);
```

Métodos

- substring (posinicial, posfinal)
- Retorna un String extraída de otro, desde el carácter 'posinicial' hasta el 'posfinal'-1:

cadena3=cadena1.substring(2,5);

En este ejemplo, "cadena3" contendrá los caracteres 2, 3, 4 sin incluir el 5 de cadena1 (Cuidado que comienza en cero)

Métodos

- indexOf (subCadena)
- Devuelve la posición de la subcadena dentro de la cadena, o -1 en caso de no estar.
- Tener en cuenta que puede retornar 0 si la subcadena coincide desde el primer carácter.

```
let nombre='Rodriguez Pablo';
let pos=nombre.indexOf('Pablo');
if (pos!=-1)
document.write ('Está el nombre Pablo en la variable nombre');
```

Métodos

- toUpperCase()
- Convierte todos los caracteres del String que invoca el método a mayúsculas:

```
cadena1=cadena1.toUpperCase();
```

- Luego de esto, cadena1 tiene todos los caracteres convertidos a mayúsculas.

Métodos

- toLowerCase()
- Convierte todos los caracteres del String que invoca el método a minúsculas:

```
cadena1=cadena1.toLowerCase();
```

- Luego de esto, cadena1 tiene todos los caracteres convertidos a minúsculas.

Ejemplo

Cargar un string por teclado y luego llamar a los distintos métodos de la clase String y la propiedad length.

```
<script>
        let cadena = prompt('Ingrese una cadena:');
        document.write('La cadena ingresada es:' + cadena);
        document.write('<br>');
        document.write('La cantidad de caracteres son:' + cadena.length);
        document.write('<br>');
        document.write('El primer carácter es:' + cadena.charAt(0));
        document.write('<br>');
        document.write('Los primeros 3 caracteres son:' + cadena.substring(0, 3));
        document.write('<br>');
        if (cadena.indexOf('hola') != -1) {
            document.write('Se ingresó la subcadena hola');
        } else {
            document.write('No se ingresó la subcadena hola');
        }
        document.write('<br>');
        document.write('La cadena convertida a mayúsculas es:' + cadena.toUpperCase());
        document.write('<br>');
        document.write('La cadena convertida a minúsculas es:' + cadena.toLowerCase());
        document.write('<br>');
    </script>
```

EJERCICIOS ADICIONALES PROPUESTOS

¡MUCHAS GRACIAS!