

# JAVASCRIPT

Guía estudiantil



Elaborado por el formador:  
Heladio Polo Castro

INSTITUTO COLOMBIANO DE APRENDIZAJE  
INCAP



**EL SIGUIENTE MATERIAL SE PREPARÓ CON FINES ESTRICTAMENTE ACADÉMICOS, DE ACUERDO CON EL ARTÍCULO 32 DE LA LEY 23 DE 1982, CUYO TEXTO ES EL SIGUIENTE:**

**ARTÍCULO 32:**

“Es permitido utilizar obras literarias o artísticas o parte de ellas, a título de ilustración en obras destinadas a la enseñanza, por medio de publicaciones, emisiones de radiodifusión o grabaciones sonoras o visuales, dentro de los límites justificados por el fin propuesto, o comunicar con propósitos de enseñanza la obra radiodifundida para fines escolares, educativos, universitarios y de formación profesional sin fines de lucro, con la obligación de mencionar el nombre del autor y el título de las obras así utilizadas.”

**(PE) Javascript**

Instituto Colombiano de Aprendizaje

**Elaborado por:**

Heladio Polo Castro

**Editado por:**

Instituto Colombiano de Aprendizaje INCAP Avenida Caracas No. 63-66

© Prohibida la reproducción parcial o total bajo cualquier forma

(Art. 125 Ley 23 de 1982)

Bogotá – Colombia

Versión 01 – enero 2021

## CONTENIDO

PRESENTACIÓN .....	6
GUÍA METODOLÓGICA .....	7
INFORMACIÓN GENERAL Y VALORACIÓN DEL PROCESO DE FORMACIÓN..	<b>¡Error! Marcador no definido.</b>
RECOLECCIÓN DE EVIDENCIAS DE APRENDIZAJE.....	<b>¡Error! Marcador no definido.</b>
UNIDAD 1.....	8
1.1 ¿Qué es JavaScript? .....	8
1.2 Como se visualiza JavaScript.....	8
1.3 Hablemos de la SINTAXIS.....	8
1.4 Estructuras básicas de programación en JavaScript .....	9
1.4.1 Variables y constantes.....	9
1.4.2 Tipos de variables y constantes.....	10
Codigo JavaScript.....	11
1.5 Comentarios en Javascript .....	15
1.6 Consola en Javascript .....	15
1.7 Strings .....	16
2 DOM (Document object Model) .....	16
Tipos de Nodos .....	19
2.1 OBJETO DOCUMENT –DOM- .....	19
2.1.1 Forma de acceder a los nodos.....	21
2.1.2 Encontrar elementos html.....	21
2.1.3 Sustituir elementos html.....	22
2.1.4 Crear y quitar elementos .....	22
2.1.5 Agregar funciones y eventos onclick con id.....	23
2.1.6 Encontrar elementos HTML por selectores CSS .....	23
2.1.6.1 Document.getElementsByTagName(): .....	23
2.1.6.2 Document.getElementsByName(): .....	26
2.1.6.3 Document.getElementById(): .....	27
2.1.7 Crear Elementos HTML .....	28
2.1.8 Como incluir JavaScript en documentos HTML.....	33
2.1.9 JavaScript definido en un documento externo .....	34
2.1.10 Incluir JavaScript en los elementos HTML .....	35
UNIDAD 2.....	38
METODOS, EVENTOS Y PROPIEDADES DE JAVASCRIPT .....	38
2.2 Propiedad innerHTML .....	38

2.3	Métodos String de Javascript .....	39
2.3.1	Método length .....	39
2.3.2	Método concat() .....	39
2.3.4	Método Substring() .....	40
2.3.5	Método Replace().....	41
2.3.6	Función toUpperCase .....	41
2.3.7	Función toLowerCase .....	42
2.4	Métodos Numéricos de Javascript.....	42
2.4.1	Método toString() .....	43
2.4.2	Método toFixed() .....	43
2.4.3	Método parseInt .....	44
2.4.4	Método isNaN .....	44
2.4.5	Método parseFloat .....	45
2.5	Métodos Arrays de Javascript.....	46
2.5.1	Método toString() .....	46
2.5.2	Método POP() .....	46
2.5.3	Método PUSH() .....	47
2.5.4	Método SHIFT() .....	48
2.5.5	Método UNSHIFT().....	49
2.5.6	Método SPLICE().....	49
2.5.7	Método MATH.....	50
2.6	Estructuras secuenciales de un programa .....	51
2.6.1	Captura de datos en Javascript .....	51
2.6.2	Salida de datos en Javascript .....	52
2.6.3	Ejemplos con instrucciones básicas .....	53
2.7	Estructuras de control de flujo de un programa.....	57
2.7.1	Transferencia de control condicional básica(if) .....	57
2.7.2	Transferencia de control condicional anidado (if/else) .....	59
2.7.3	Ciclos repetitivos o Bucles: .....	61
2.7.3.1	Ciclo for():.....	61
2.7.3.2	Ciclos while.....	63
2.7.3.3	Estructura switch: .....	64
2.8	Programación avanzada.....	66
2.8.1	Funciones Personalizadas .....	66
2.9	Eventos en JavaScript.....	69
2.10	Barra menú lateral Colapsada .....	72
2.11	Ventana Modal con funciones JS .....	74

2.12	Método AddEventListener()	79
2.13	Funciones anidadas Método AddEventListener()	82
2.14	Objetos en Javascript	83
2.14.1	Propiedades y métodos del objeto	83
UNIDAD 3		87
FORMULARIOS Y VENTANAS DE DIALOGO EN JAVASCRIPT		87
2.15	Formularios en JavaScript	87
2.16	Manejo de Controles visuales en un formulario	87
2.16.1	Cuadro de texto y textarea:	87
2.16.2	Botones de opción (radiobutton):	90
2.16.3	Casillas de verificación (checkbox):	91
2.16.4	Cuadros combinados (select):	91
2.17	Validación de un formulario en JS	92
2.17.1	Validación de campos obligatorios:	92
2.17.2	Validación de campos numéricos:	92
2.17.3	Validación de campos de correo electrónico:	93
2.17.4	Validación de cuadros combinados:	93
2.17.5	Validación de casillas de verificación (checkbox):	93
2.17.6	Validación de campos tipo fecha:	94
2.17.7	Validación de botones de opción (radio button):	94
2.17.8	VALIDAR FORMULARIO BASICO	95
2.18	EXPRESIONES REGULARES EN JS	97
2.18.1	METODOS DE EXPRESIONES REGULARES	97
2.18.2	Patrones de búsquedas de expresiones regulares	98
2.18.3	Validación avanzada de formularios	100
3	FRAMEWORK (LIBRERÍAS) EN JS	111
3.1	Qué son las librerías	111
3.1	VISOR DE IMÁGENES CON JS	112
3.2	JQuery	116
3.2.1	EVENTO DOCUMENT READY()	119
3.2.2	METODO TEXT()	120
3.2.3	METODO HTML()	121
3.2.4	EVENTO CLICK y METODO CSS()	122
3.2.5	METODO CLASS()	124
3.2.6	EVENTOS TOGGLE()	126
3.3	Jquery con @media Queries y Side Bar Left	127
GLOSARIO		138



## PRESENTACIÓN

En la actualidad, la tecnología avanza de manera acelerada y es importante resaltar que uno de los oficios que se vuelve rentable económicamente para cualquier persona es el de programador o desarrollador de software; con esta profesión podemos cambiar el mundo tener una perspectiva mas amplia, poder pensar de forma analítica y sobre todo tener mente creativa es inevitable referirnos a que no todas las personas se atreven a incursionar en este campo, ya sea por miedo a lo complejo que resulte ser o por la falta de conocimientos y, porque no, a la tradicional pereza de utilizar un poquito de su intelecto en el desarrollo de nuevas soluciones informáticas.

Este lenguaje que es uno de los más utilizado en el desarrollo de aplicaciones Web dinámicas en el mundo, es también uno de los más sencillos y fáciles de aprender a manejar, convirtiéndose en una alternativa para perder el miedo al desarrollo de software y crear aplicaciones de manera eficiente, que ofrezcan la oportunidad de recibir buenos dividendos de una manera fácil y con recursos básicos (un buen computador en casa y muchas ganas y motivación).

Actualmente este lenguaje direccionado a la web podemos aplicar importantes funcionalidades que sirvan para dar solución a una necesidad en ambiente laboral o comercial. Aprovechenlo al máximo y no olvide que aquellos que se atreven a soñar y a llevar a cabo sus sueños, se convertirán tarde que temprano en los dueños del mundo.

## GUÍA METODOLÓGICA

La estrategia metodológica del INCAP para la formación técnica del aprendiz mediante competencias laborales, comprende dos caminos:

1. Las clases presenciales dictadas por el Formador: haciendo uso del método Inductivo – Activo.
2. El trabajo práctico de los estudiantes: dirigido y evaluado por el Formador, a través de talleres, desarrollo de casos, lecturas y consultas de los temas de clase, etc. Con esto se busca fomentar en el estudiante el análisis, el uso de herramientas tecnológicas y la responsabilidad.

Los módulos guías utilizados por el INCAP, para desarrollar cada uno de los cursos, se elaboran teniendo en cuenta una metodología con características y recomendaciones de uso:

- A cada unidad de aprendizaje le corresponde un logro de competencia laboral, el cual viene definido antes de desarrollar su contenido.
- Seguidamente, se definen los indicadores de logro, que son las evidencias de aprendizaje requeridas que evaluará el Formador.
- Desarrollo de la unidad dividida en contenidos breves y ejercicios, referenciados así: FDH (Formador Dice y Hace): corresponde a la explicación del contenido y el desarrollo de los ejercicios por parte del Formador.

FDEH (Formador Dice y el Estudiante Hace): el estudiante desarrolla los ejercicios propuestos y el Formador supervisa.

EDH (Estudiante Dice y Hace): es el trabajo práctico que desarrollan los estudiantes fuera de la clase, a través de talleres, desarrollo de casos, lecturas y consultas de los temas, los cuales deben ser evaluados por el Formador.

- Al final de cada unidad, se puede encontrar un resumen de los contenidos más relevantes y de los ejercicios generales.
- Glosario: definición de términos o palabras utilizadas en la unidad que son propias del tema a tratar.





## FDH. (EL FORMADOR DICE Y HACE)

### 1.1 ¿Qué es JavaScript?

JavaScript es un lenguaje interpretado (no requiere de un compilador para ser ejecutado) que se utiliza tradicionalmente en páginas web dinámicas; este lenguaje nos permite incorporar interactividad de las páginas con el usuario de las mismas; por ejemplo, texto que desaparece, animaciones, acciones que se activan al dar clic sobre un botón, etc.

Se debe aclarar que este lenguaje de scripts nada tiene que ver con el lenguaje JAVA y no se debe confundir con él; tienen algunas similitudes, pero no son lo mismo y ninguno depende del otro.

Para programar código se utiliza entre las etiquetas `<script>` `</script>`

### 1.2 Como se visualiza JavaScript

JavaScript tiene diferentes formas de salida de datos, así:

- ✓ Escriba en un elemento HTML contenido diverso, utilice **innerHTML**.
- ✓ Muestre en el navegador web, contenido HTML utilice **document.write()**.
- ✓ Muestre un mensaje en un cuadro de dialogo tipo alerta, utilice **window.alert()**.
- ✓ Escriba un mensaje en la consola del navegador, use **console.log()**.

### 1.3 Hablemos de la SINTAXIS

En JavaScript como en los demás lenguajes de programación, se deben seguir ciertas reglas para escribir los comandos o líneas de programación. Aquí describimos algunas de estas reglas que permitirán realizar Scripts bien estructurados y guardando o respetando las características técnicas del lenguaje:

- a) No se tienen en cuenta los espacios en blanco ni los saltos de línea y esto nos permite tabular las líneas de código para establecer jerarquías y de esta manera comprenderlo mejor.
- b) Es sensible a MAYUSCULAS y minúsculas, es decir que para que funcione adecuadamente un Script debemos utilizar de manera adecuada tanto mayúsculas como

minúsculas, sobre todo al definir variables u objetos sobre los cuales se creara código de programación que los pueda llegar a afectar.

c) No se requiere establecer el tipo de dato que se almacenara en las variables y esto nos permite definir una variable y guardar en ella diferentes tipos de datos durante la ejecución de un Script sin que JavaScript reporte errores por este hecho.

d) Aunque se recomienda terminar cada línea de programación con un punto y coma (;), en JavaScript no es necesario ni se obliga.

e) Se pueden utilizar comentarios para documentar un programa y además se recomienda el uso de los mismos para dar legibilidad y claridad a los programas escritos. Esta es una buena costumbre que recomiendo no abandonar. Para comentar una sola línea se utiliza // al principio de la línea y para comentar varias líneas utilizamos /\* al inicio y \*/ al final de la última línea de comentario.

## 1.4 Estructuras básicas de programación en JavaScript

### 1.4.1 Variables y constantes

Una variable como todos sabemos, es un espacio de memoria cuyo contenido puede variar dentro o después de la ejecución de un programa; al nombre que se le da a dicha variable también se le puede denominar **identificador**. Es importante destacar que las variables hacen más recursivos y eficientes los programas y que los identificadores en todo lenguaje de programación se rigen por ciertas reglas que recordaremos a continuación:

a) No deben contener caracteres especiales ni espacios en blanco y no deben empezar por un valor numérico.

b) Debe ser lo suficientemente significativo que permita de manera lógica determinar qué tipo de dato se almacenara en la variable.

c) No debe ser ni muy extenso ni muy corto; debe tener una longitud razonable que permita su fácil comprensión.

d) El único carácter especial que se permite para un identificador es el guion bajo, raya baja o raya al piso (\_).

e) **JavaScript distingue en los identificadores entre mayúsculas y minúsculas.**

Normalmente las variables en JavaScript se definen utilizando la **palabra reservada** var tal como se ve en los siguientes ejemplos:

```
var numero1=3;
```

```
var numero2=5;
```

```
var resultado=numero1+numero2;
```

```
const mes=1;
```

Observemos que cada línea de programación termina con un punto y coma (;) situación que

ya habíamos mencionado anteriormente. Se aclara que las variables solo son definidas una sola vez, es decir que la palabra reservada `var` solo se utiliza cuando se definen por primera vez las variables, luego no es necesario utilizarla. Pero recordemos que JavaScript es un lenguaje que no obliga a definir variables antes de utilizarlas, ya que se pueden utilizar en el momento que se requieran y aunque esta es una práctica que no se recomienda, tratando de respetar los buenos hábitos o las buenas prácticas en programación.

#### 1.4.2 Tipos de variables y constantes

**Numéricas:** nos permiten guardar información de tipo numérico (Enteros, decimales, etc).

**const iva=19;**

**var Estatura=1.85;**

Como podemos observar los valores decimales **utilizan como separador el punto (.)** y no la coma (,) en JavaScript.

**Cadena de caracteres:** utilizadas para almacenar texto, caracteres, palabras, etc. Para asignar un valor textual a una variable este se debe encerrar entre comillas dobles ( " ") o simples ( ' ') según el caso.

**var mensaje=" Bienvenidos al INCAP" var Nombre\_Producto= "Disco Duro"**

Cuando se requiere de incluir caracteres especiales que generen algún tipo de problemas es necesario utilizar unas combinaciones de caracteres que a continuación se describen:

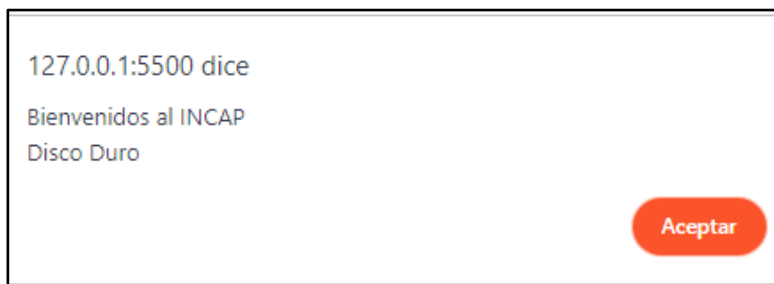
Incluir	Combinación de caracteres
Una Nueva línea	\n
Un tabulador	\t
Una comilla simple	\'
Una comilla doble	\"
Una barra inclinada	\\

Veamos un ejemplo de variables inicializadas con datos de tipo cadena de caracteres y numéricos:

## Codigo JavaScript

```
<body>
<script>
//Asignación de variables tipo texto
var mensaje="Bienvenidos al INCAP";
var Nombre_Producto="Disco Duro";
//Salida de resultados de variables
alert( mensaje + "\n" + Nombre_Producto);
document.write(mensaje + "<br>" + Nombre_Producto);
</script>
</body>
</html>
```

## Resultado



### 1.4.3 Arrays

También llamados vectores, matrices o arreglos, me permiten agrupar en una sola variable una colección de elementos que pueden ser del mismo tipo o de tipos diferentes.

Supongamos que queremos realizar algún tipo de procedimiento con los días de la semana; para ello podríamos crear siete variables para almacenar cada día de la semana y trabajar con cada uno de ellos; pero si en lugar de crear siete variables se crea una única variable que me permita guardar en ella los siete días de la semana y poder trabajar con cada uno de ellos independientemente, esto nos permitiría crear un código de programación más limpio, eficiente y dinámico.

Para declarar una variable tipo array, se utilizan los caracteres [...] y su índice de posición

comienza desde cero (0), luego 1 , 2 ,3, .... n dependiendo el tamaño del mismo, de la siguiente manera:

```
var dias_semana=["Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"]  
    posición: [ 0 , 1 , 2 , 3 , 4 , 5 , 6 ]
```

Así, entonces si quisiera acceder o mostrar el primer día de la semana que es ***lunes***, se podría utilizar el siguiente código:

Ejemplo 1:

```
<body>  
<script>  
var dias_semana=["Lunes","Martes","Miercoles","Jueves","Viernes","Sabado","Domingo"]  
// En este ejemplo accedemos a la posición 0 del array que será el día lunes  
var PrimerDia=dias_semana[0];  
alert (PrimerDia);  
console.log(PrimerDia);  
</script>  
</body>  
</html>
```

Resultado

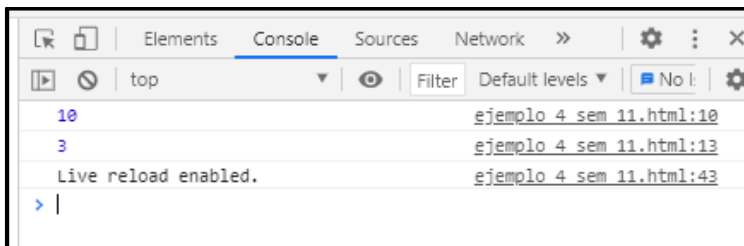


**Ejemplo 2:**

```
<body>  
<script>  
var arreglo=[20,10,5]  
// En este ejemplo accedemos a la posición 1 del array que será el valor 10  
console.log(arreglo[1]);  
alert(arreglo[1]);
```

```
// Se muestra por consola la longitud o tamaño del array que es 3 elementos
console.log(arreglo.length);
</script>
</body>
</html>
```

Resultado:



Recuerde que: para acceder a cada elemento solo basta con indicar el índice o la posición del mismo, teniendo en cuenta que los índices o posiciones en programación casi siempre empiezan en 0.

#### 1.4.4 Objetos en JS

Los objetos en Javascript son variables que pueden almacenar datos de diferentes características.

```
var producto = {categoria:"Silla", marca:"rimax", color:"blanca"};
```

Para acceder a las propiedades del objeto producto se haría de la siguiente forma:  
*Nombreobjeto.nombrepropiedad como p. ejemplo: producto.marca y producto.color*

### 1.4.5 Operadores:

Los operadores permiten la manipulación de los valores almacenados en las variables y así verdaderamente sacarle el mayor provecho a la programación; por eso a continuación describimos de manera básica cada una de las clases de operadores:

De Asignación		
=	Me permite crear un incremento fijo de uno en uno a una variable	<code>var contador=contador + 1;</code>
+=	Permite crear un acumulador de valores numéricos	<code>Tot += edad;</code>
De Incremento y Decremento		
++	Incrementa en uno el valor de una variable	<code>var Valor1=5; ++Valor1; //Valor1 es ahora 6</code>
--	Decrementa en uno el valor de una variable	<code>var Valor1=5; --Valor1; //Valor1 es ahora 4</code>
Lógicos		
! (Negación)	Se utiliza para obtener el valor contrario de la variable.	<code>var Cantidad=0; !Cantidad; // Cantidad=true Var Cantidad=2; !Cantidad; //Cantidad=false</code>
&& (AND)	Está relacionado con la tabla de verdad And y obtiene su resultado de la combinación de dos valores booleanos.	<code>var Valor1=true; var Valor2=true; Resultado=Valor1&amp;&amp;Valor2; //Resultado es true</code>
(OR)	Está relacionado con la tabla de verdad Or y obtiene su resultado de la combinación de dos valores booleanos.	<code>var Valor1=true; var Valor2=false; Resultado=Valor1   Valor2; //Resultado es true</code>
Matemáticos		

+ , - , * , / , %(Modulo)	Me permiten generar las diferentes operaciones matemáticas básicas. Una forma de simplificar las expresiones matemáticas es utilizar el operador antes del signo igual: +=, -=, *=, /=	<b>var Valor1=5; var Valor2=2;</b> <b>Mult=6;</b> <b>Suma=Valor1+Valor2;</b> <b>Resta=Valor1-Valor2;</b> <b>Residuo=Valor1%Valor2;</b> <b>Mult*=4;</b>
<b>De Relación</b>		

> (Mayor que) < (Menor que) >= (Mayor o igual) <= (Menor o igual) == (Igual, comparación) != (Diferente)	Estos operadores nos permiten comparar dos valores y su resultado siempre será un resultado lógico el cual puede ser falso o verdadero, según el caso. Es importante anotar que cuando se comparan cadenas de caracteres, las que anteceden son menores a las que preceden y las mayúsculas son menores que las minúsculas.	<b>Var Num1=2; Var Num2=6;</b> <b>Comp=Num1&gt;Num2;</b> <b>// Comp quedara con false</b> <b>Comp2=Num1!=Num2;</b> <b>// Comp2 quedara con true</b> <b>Comp3=Num1==Num2;</b> <b>// Comp3 quedara con false</b> <b>Comp4=Num1&lt;Num2;</b> <b>// Comp4 quedara con true</b>
---	---	--

## 1.5 Comentarios en Javascript

Los comentarios en JavaScript tienen dos utilidades:

- ✓ Para explicar el código JavaScript y así poder documentar y entenderlo más fácilmente.
- ✓ Para evitar la ejecución de un fragmento de código de forma eventual.
- ✓ Los comentarios se escriben comenzando por dos símbolos: //

## 1.6 Consola en Javascript

La consola, es una pestaña que integra el inspector de elementos (pestaña *element*) que es una sección del navegador web donde tenemos control a varios aspectos referentes a la página actual visualizada como, por ejemplo: su estructura de etiquetas HTML, sus estilos CSS, sus scripts, entre otros, en la pestaña **Console** podemos visualizar los resultados de Javascript utilizando la sentencia *console.log*, también podemos revisar resultados de variables, ejecutar funciones o sentencias de Javascript.



### Atajos de teclado:

1. Para ir al Inspector de elementos CTRL + SHIFT + I
2. Para abrir a la pestaña consola: CTRL + SHIFT + J o TECLA F12
3. Para borrar la consola: CTRL + L

### 1.7 Strings

Los strings en JavaScript se declaran para almacenar cadenas de textos.

Ejemplo

```
Var marca="huawei";
```

### FDEH

- ✓ Estudiante en el siguiente ejercicio debes de completar las siguientes definiciones:  
Cuáles son las diferencias entre utilizar variables y contantes

---

---

---

- ✓ Las variables en Javascript necesitan definir su tipo de datos a utilizar sí o no y porque?

---

---

## 2 DOM (Document object Model)

Esta novedad en JavaScript les permite a los programadores acceder y manipular a su antojo las paginas HTML. Este lenguaje se ha convertido en una utilidad disponible para la gran mayoría de lenguajes de programación, entre ellos PHP, Java, JavaScript, etc.

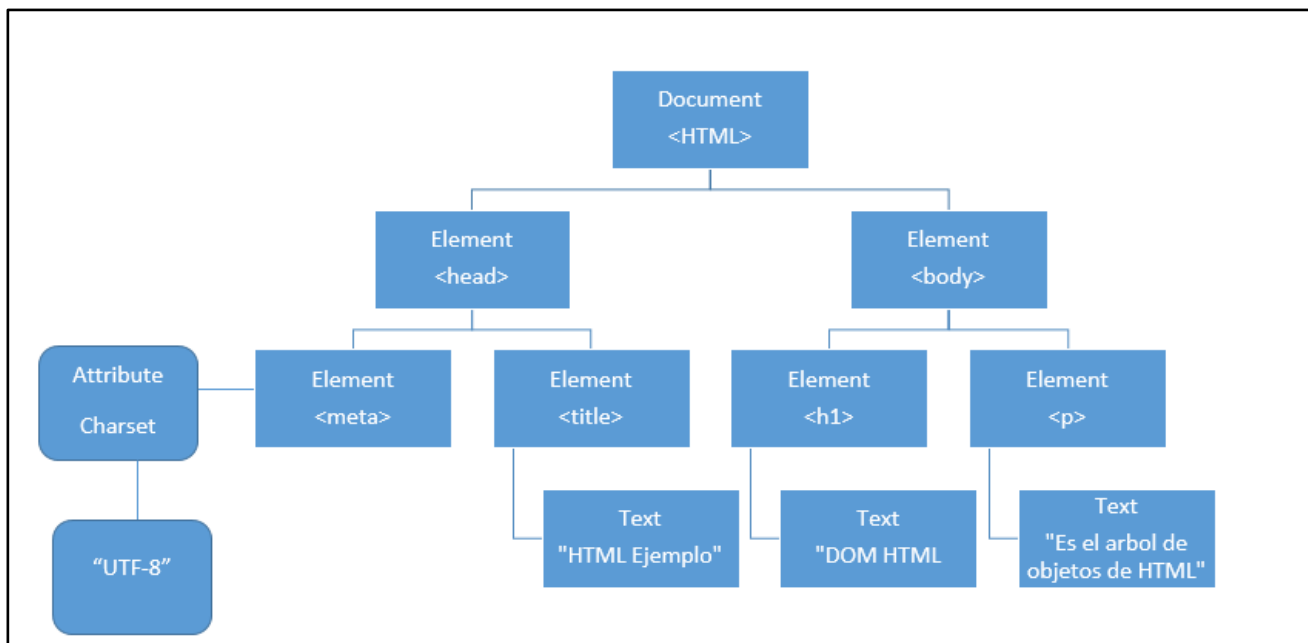
Algunas de las tareas que se facilitan aplicando las ventajas de DOM son entre otras: obtener un valor almacenado por algunos elementos de un formulario; crear un elemento (párrafo <p>, <div>, etc.) de manera dinámica y poderlo añadir a una página; añadir movimiento o animación a un elemento o hacer que aparezca o desaparezca. Para poder sacarle todo el provecho a DOM se requiere realizar una trasformación en la página HTML ya que esta es una sucesión de caracteres, característica que hace difícil su manipulación; esta trasformación la mayoría de navegadores la realizan de manera automática aplicándoles una estructura más eficiente a las paginas HTML para manipular.

DOM transforma todos los elementos de una página HTML en un conjunto de elementos

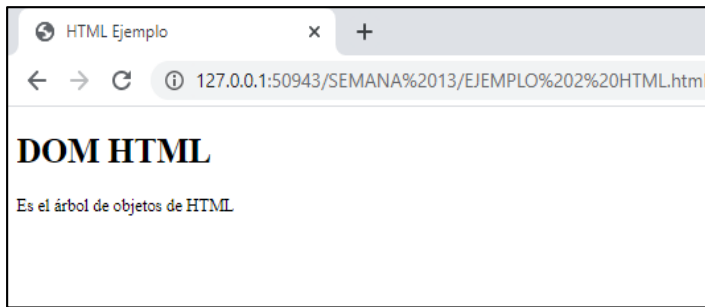
llamado Nodos los cuales se encuentran interconectados formando un árbol de Nodos, y al igual que un árbol vamos a encontrar ramificaciones determinando de esta manera jerarquías. Miremos un ejemplo:

```
<html>
<head>
  <title>HTML Ejemplo</title>
  <meta charset="UTF-8">
</head>
<!-- En esta estructura básica de DOM HTML vemos que el nodo principal <HTML> luego
siguen los nodos <head> y <body> y después de esos nodos vendrán los subnodos
respectivos. Ver imagen de árbol de nodos - - >
<body >
<h1>DOM HTML</h1>
<p>
Es el árbol de objetos de HTML
</p>
</body>
</html>
```

La página anterior la podemos trasladarla al siguiente árbol de Nodos:



## RESULTADO FINAL:



Aunque el ejemplo maneja una página HTML muy sencilla, esta se utilizó para entender de manera igualmente sencilla el concepto del DOM.

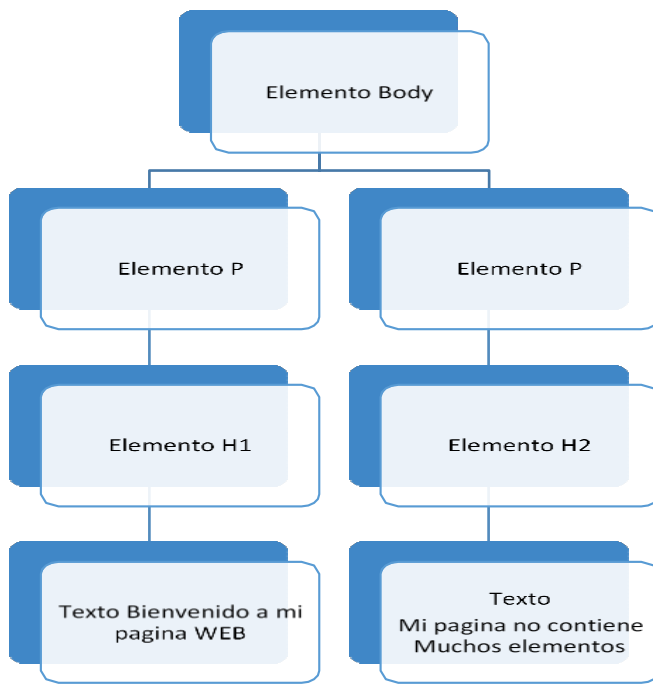
La transformación automática de una página HTML en un árbol de nodos siempre sigue las mismas reglas:

- Las etiquetas HTML se transforman en dos nodos: el primero corresponde a la propia etiqueta y el segundo nodo es hijo del primero y normalmente corresponde al texto contenido en la primera etiqueta o primer nodo.
- Si una etiqueta HTML se encuentra dentro de otra, se realiza el mismo proceso anterior, con la diferencia que los nodos resultantes serán hijos de su etiqueta padre.

Un ejemplo de lo anterior sería:

```
<body>
  <p>
    <h1>Bienvenido a mi página WEB.</h1>
  </p>
  <p>
    <h2>Mi pagina no contiene muchos elementos.</h2>
  </p>
</body>
```

Y este sería el árbol de Nodos:



### Tipos de Nodos

Aunque podemos encontrar 12 tipos de nodos, mencionaremos los cinco principales mas no los únicos y recalco lo anterior mas no los únicos:

- a) **Document**: nodo raíz del cual se desprenden todos los demás.
- b) **Element**: representa cada una de las etiquetas HTML, este nodo es el único que puede contener atributos y del cual se desprenden otros nodos.
- c) **Attribute**: se define para representar cada uno de los atributos de las etiquetas HTML.
- d) **Text**: nodo contenedor del texto definido en una etiqueta.
- e) **Comment**: representa los comentarios incluidos en una página HTML.

### 2.1 OBJETO DOCUMENT –DOM-

El objeto document es el propietario de todos los demás objetos de su página web, El objeto document representa toda la página web. Si desea acceder a un determinado elemento ubicado en una página HTML, siempre comience por colocar al objeto document.

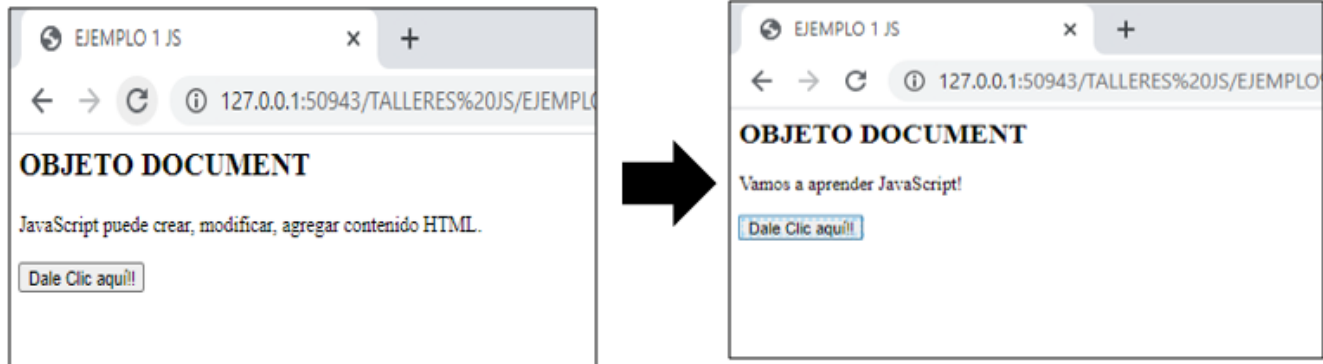
*Ejemplo 1: con evento onclick*

```
<html>
<body>
<title>EJEMPLO 1 JS</title>
<h2>OBJETO DOCUMENT</h2>
```

```

<p id="docto">JavaScript puede crear, modificar, agregar contenido HTML.</p>
<button type="button" onclick='document.getElementById("docto").innerHTML =
"Vamos a aprender JavaScript!"> Dale Clic aquí!!</button>
</body>
</html>

```



## Ejemplo 2

```

<html>
<body>
<title>EJEMPLO 1 JS</title>
<h2>OBJETO DOCUMENT</h2>
<p id="docto">JavaScript puede crear, modificar, agregar contenido HTML.</p>
<script>
document.write("Vamos a aprender JavaScript!");
</script>
</body>
</html>

```



## FDEH (Formador dice estudiante hace)

Estudiante para reafirmar este conocimiento debes de realizar el árbol de nodos de este documento HTML

```
<html>
<head>
  <title>PAGINA WEB INICIAL</title>
  <meta charset="UTF-8">
</head>
<body bgcolor="lightblue">
<p><font color="red">PHP - JAVA - JAVASCRIPT- HTML- C++</font></p>
</body>
</html>
```

### 2.1.1 Forma de acceder a los nodos

DOM permite acceder de dos formas diferentes a un nodo: el acceso a través de su nodo padre o de manera directa.

La primera forma es poco recomendable y es por esta razón que nos dedicaremos a explicar las formas en las que puedo acceder directamente a los diferentes nodos, sin necesidad de hacer todo el recorrido desde el nodo raíz, hasta el nodo buscado. Cabe anotar que el árbol de nodos es construido en su totalidad solo cuando la página HTML es cargada por completo y solo en este momento se pueden realizar las modificaciones pertinentes a los nodos respectivos.

### 2.1.2 Encontrar elementos html

Metodo Javascript	Que hace
<code>document.getElementById(<i>id</i>)</code>	Buscar un elemento por su identificador id
<code>document.getElementsByTagName(<i>nombre</i>)</code>	Busca elementos por su etiqueta, este método genera un array

<code>document.getElementsByName(<i>nombre</i>)</code>	Buscar elementos por su clase name, este método genera un array
--	---

### 2.1.3 Sustituir elementos html

Metodo Javascript	Que hace
<code>element.innerHTML = <i>Nuevo Contenido html</i></code>	Cambia el Contenido de un element HTML
<code>element.attribute = <i>Nuevo valor</i></code>	Cambia el valor del atributo de un element HTML
<code>element.style.propiedad = <i>Nuevo estilo</i></code>	Cambia el estilo de un elemento HTML
Metodo Javascript	Que hace
<code>element.setAttribute(<i>atributo, valor</i>)</code>	Cambia el valor del atributo de un element HTML

### 2.1.4 Crear y quitar elementos

Metodo Javascript	Que hace
<code>document.createElement(<i>elemento</i>)</code>	Crear un elemento HTML
<code>document.removeChild(<i>elemento</i>)</code>	Eliminar un elemento HTML

<code>document.appendChild(<i>elemento</i>)</code>	Agregar un elemento HTML
<code>document.replaceChild(<i>nuevo, anterior</i>)</code>	Reemplaza un elemento HTML
<code>document.write(<i>texto</i>)</code>	Escribe en el navegador web texto indicado

### 2.1.5 Agregar funciones y eventos onclick con id

Metodo Javascript	Que hace
<code>document.getElementById(<i>id</i>).onclick = function(){<i>codigo</i>}</code>	Crea código Html en instrucciones para programar un evento de tipo onclick

### 2.1.6 Encontrar elementos HTML por selectores CSS

Metodo Javascript	Que hace
<code>document.querySelector("nombre clase")</code>	Busca todos los estilos del primer elemento CSS que cumplan el criterio de coincidencia con un selector CSS clase especificado
<code>document.querySelectorAll("elemento nombre clase")</code>	Busca todos los elementos HTML que cumplan el criterio de coincidencia con un selector CSS aplicado (id, nombres de clase)

#### 2.1.6.1 Document.getElementsByTagName():

Este método `getElementsByTagName` (Nombres de Etiquetas) se utiliza para buscar y referenciar a todos los elementos de la página HTML que coincidan con el nombre de la etiqueta establecido dentro del paréntesis; este nombre de etiqueta es el parámetro que se pasa en la función.

La siguiente línea de script hace referencia a todos los párrafos `<p>` de la página HTML:

```
var parrafos= document. getElementsByTagName("p");
```

En el DOM `document` es el nodo principal a partir del cual se realiza la búsqueda de los elementos. El valor devuelto por la función es un **array** con todos los nodos `<p>` que cumplen

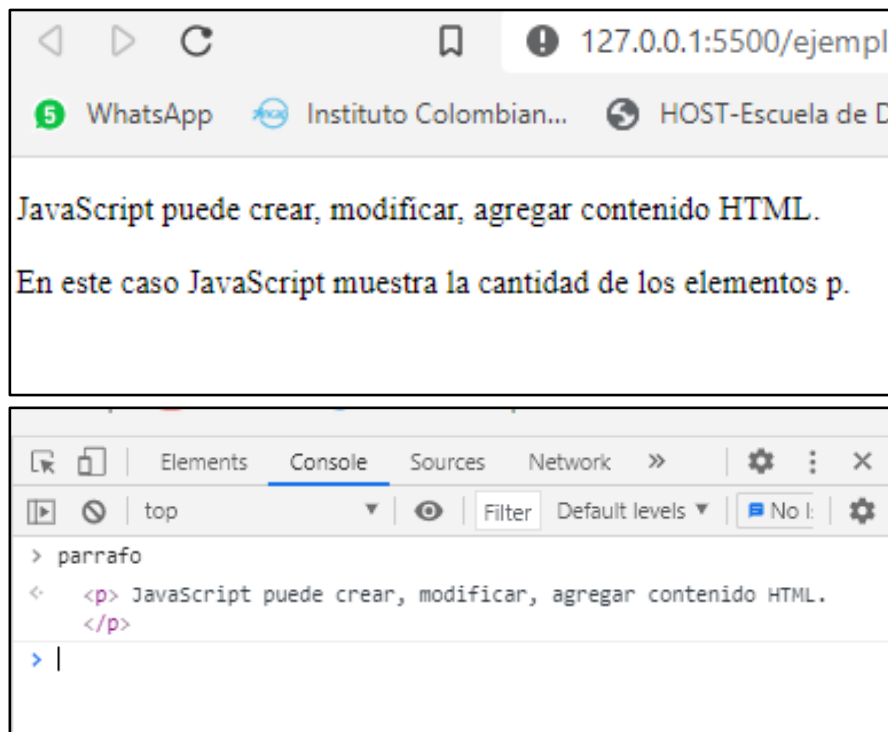


con la condición establecida en el parámetro que se pasa en la función. Dicho array es de nodos y no de texto, situación que obliga a procesar los valores del array de la siguiente manera:

Ejemplo:

```
<p> JavaScript puede crear, modificar, agregar contenido HTML.</p>
<p>En este caso JavaScript muestra la cantidad de los elementos p.</p>
<script>
  var parrafos= document. getElementsByTagName("p");
  var parrafo= parrafos[0];
  //El código anterior nos devuelve el primer párrafo de la página HTML
  document.innerHTML = parrafo;
</script>
```

## RESULTADO FINAL



El código anterior recorre todos los párrafos contenidos en la página HTML y el resultado lo almacena en la variable parrafo.

## EJEMPLO 2

```

<!DOCTYPE html>
<html>
<title>EJEMPLO 2 JS</title>
<body>
<h2>OBJETO DOCUMENT</h2>
<p> JavaScript puede crear, modificar, agregar contenido HTML.</p>
<p>En este caso JavaScript muestra la cantidad de los elementos p.</p>
<script>
// Se almacena en la constante parrafos el array de elementos p, se declara la constante
parrafo de tipo array, se genera un bucle for para leer la cantidad de los elementos <p> que
están en documento HTML, luego muestra en navegador, los parrafos y el objeto parrafo
html del array, el código anterior nos devuelve el primer párrafo de la página HTML

const parrafos= document. getElementsByTagName("p");
  const parrafo= parrafos[0];
    for (let i=0; i<parrafos.length; i++) {
      const parrafo=parrafos[i];
      parrafo.innerHTML+= parrafos[i];
    }
</script>
</body>
</html>

```

## RESULTADO FINAL



### 2.1.6.2 Document.getElementsByName():

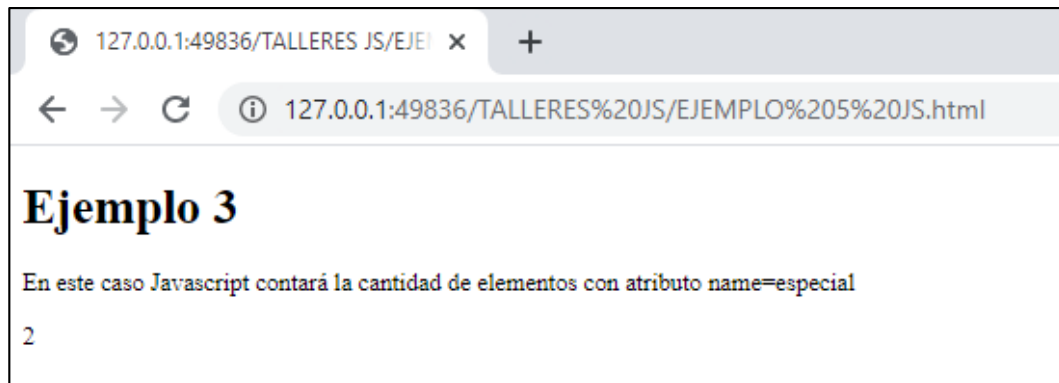
El método `getElementsByName ()` se utiliza para buscar elementos del documento HTML con un nombre especificado (atributo `name`), devuelve un array con todos los elementos, el array se guarda en el objeto **HTMLCollection**.

El objeto **HTMLCollection** contiene un array de elementos o nodos. Podemos acceder a esos nodos mediante los números de índices de su array. El índice comienza en 0.

Como por ejemplo el código que se muestra a continuación:

```
<!DOCTYPE html>
<html>
<body>
<h1 name="especial">Ejemplo 3</h1>
<p name="especial">En este caso Javascript contará la cantidad de elementos con atributo
name=especial </p>
<script>
// Se crea la variable nombre para buscar y almacenar en la cantidad de elementos con atributo
name = "especial", dentro de un array, luego muestra en navegador, la cantidad de elementos
almacenados en la variable nombre
var nombre= document.getElementsByName("especial").length;
document.write(nombre);
</script>
</body>
</html>
```

### RESULTADO FINAL



### 2.1.6.3 Document.getElementById():

Este método podemos considerarlo el que es más utilizado de todos ya que nos permite ~~hacer~~ acceder de manera directa a los nodos id de la página HTML y de esta manera leer y modificar sus propiedades.

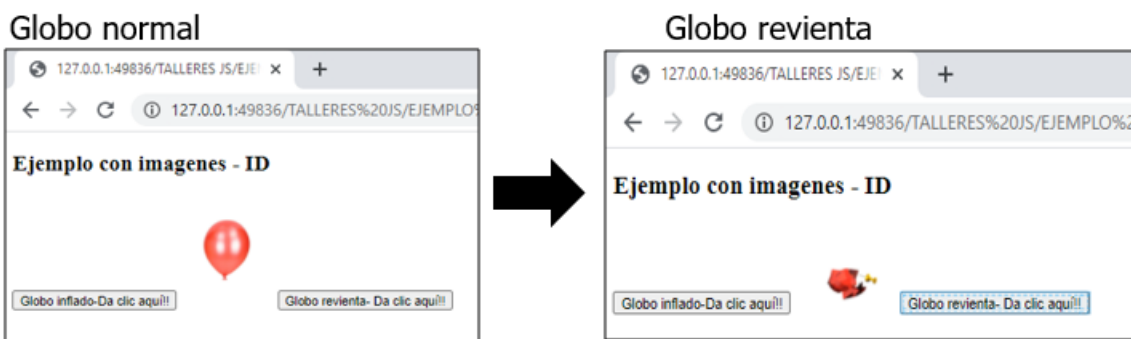
Este método devuelve el nodo cuyo Id coincida con el que se indica en el parámetro de la función, permitiendo hacer únicos cada uno de los nodos de la página HTML.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<body>
<h2>Ejemplo con imagenes - ID</h2>
<!-- Se crea dos eventos: un evento onclick para que al dar clic en el botón Globo inflado
aparezca el globo normal y otro evento onclick para que al dar clic en botón Globo revienta
aparezca reventado por un pincho -->
<button onclick="document.getElementById('globo').src='img/globo-rojo.jpg'">Globo inflado-Da clic aquí!!</button>

<button onclick="document.getElementById('globo').src='img/globo-explota.gif'">Globo revienta- Da clic aquí!!</button>
</body>
</html>
```

### RESULTADO FINAL



## FDEH

- ✓ Utilice el método `getElementById` para encontrar el elemento que tiene como identificador "HTML" y cambie su texto a "Incap la via del progreso".

```
<p id="HTML"></p>
```

```
<script>
```

```
_____ ="Incap";
```

```
</script>
```

- ✓ Utilice el método `getElementsByTagName` para encontrar el primer elemento `<p>` y cambie su texto a "Incap la via del progreso".

```
<p name="HTML"></p>
```

```
<p name="HTML"></p>
```

```
<script>
```

```
_____ ="Incap";
```

```
</script>
```

### 2.1.7 Crear Elementos HTML

Para crear elementos HTML requerimos de las siguientes funciones DOM:

- `createElement` (Etiqueta HTML): esta función crea un nodo de tipo element, determinado por la etiqueta que se pasa como parámetro en dicha función.
- `createTextNode` (contenido): esta función crea un nodo de tipo text almacenando en él el contenido textual de los elementos HTML.
- `parent.appendChild` (nodoHijo): determina un nodo como hijo de otro y funciona de la siguiente manera: en primer lugar, se debe añadir el nodo text como hijo del nodo element y luego se añade el nodo element como hijo del nodo que representa el elemento de la página HTML.

## EJEMPLO

```
<html>
<head>
<meta charset="UTF-8">
</head>
```

```

<body>
<h1 id="cabecera"></h1>
<ul id="mes"></ul>

<script type="text/javascript">
// En este ejemplo aplicamos la creación de elementos y nodos de texto, se declaran las
constantes para asignar los <id> de elementos <h1>y<ul>, luego se crea el elemento <li>
mediante document.createElement, luego se agrega al elemento <li> el nodo de texto Enero,
luego se agrega al elemento padre <ul> el mes de Enero. Por último, se imprime en pantalla
un título <h1> y se llama a la consola para mostrar var lista
const cabecera=document.getElementById("cabecera");
const mes=document.getElementById("mes");
const lista=document.createElement("li");

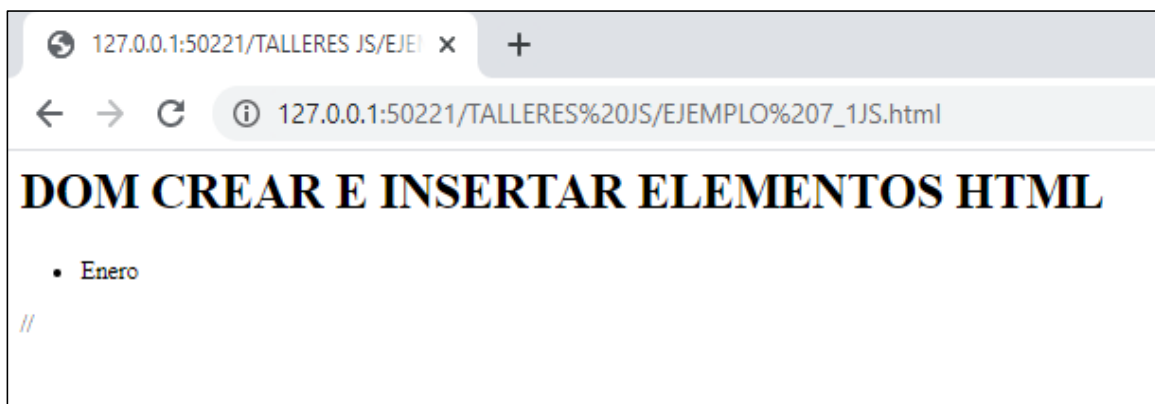
lista.textContent='Enero';
mes.appendChild(lista);

cabecera.innerHTML='DOM CREAR E INSERTAR ELEMENTOS HTML';
console.log(lista);

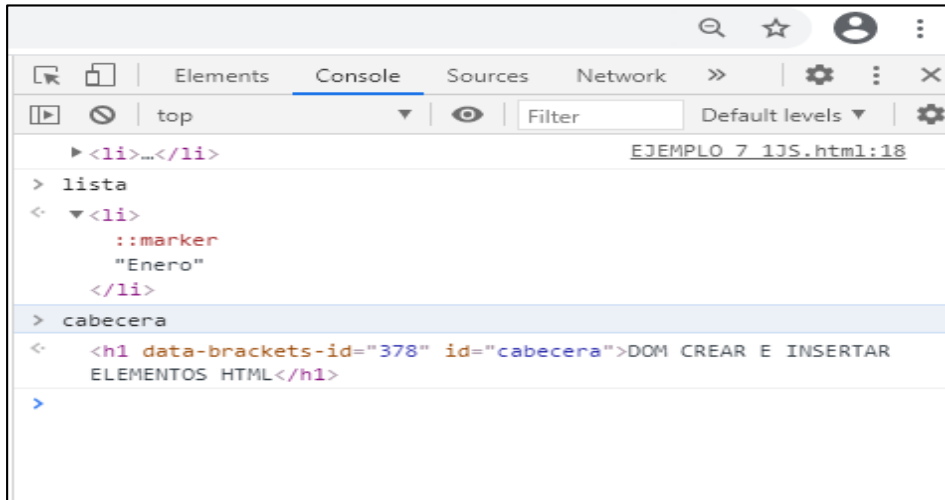
</script>
</body>
</html>

```

RESULTADO FINAL



## Ventana consola



Miremos el caso de si se quiere añadir un encabezado simple dentro de una página HTML, el script que debería utilizarse es el siguiente:

### EJEMPLO

Vamos a crear un nodo de texto y agregar a un nodo de `<ol>` un elemento `<li>`

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Ejemplo 3 DOM</title>
</head>
<body>
<ol id="lista">
  <li>Crear Elemento</li>
  <li>Agregar Elemento</li>
</ol>
```

```
<script type="text/javascript">
```

// En este ejemplo aplicamos la creación de elementos y nodos de texto, se declaran las variables: elementoli para crear el elemento <li>, mediante document.createElement y luego se crea la variable contexto para crear el nodo de texto "Nuevo texto", luego se agrega al elemento padre <li>; Por último, se busca y define el elemento padre de primer <li> con parentNode y luego se asigna el nodo de texto al elemento padre.

```
var elementoli=document.createElement("li"), contexto=document.createTextNode("Nuevo  
Texto");
```

```
elementoli.appendChild(contexto);
```

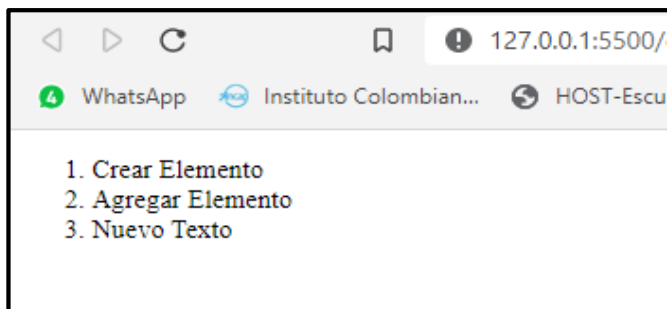
```
var padre=document.getElementsByTagName("li")[0].parentNode;
```

```
padre.appendChild(elementoli);
```

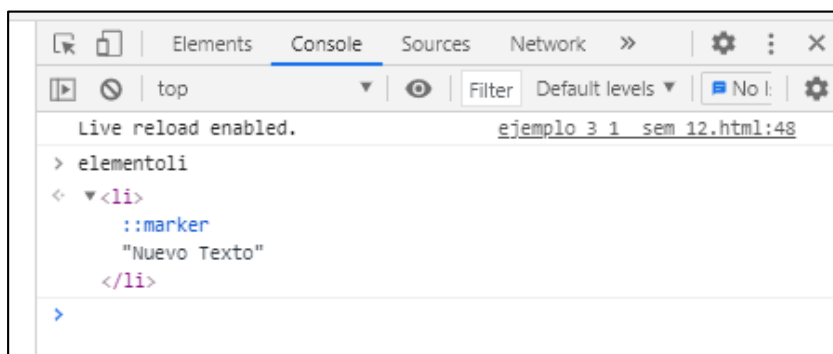
```
</script>
```

```
</body>
```

```
</html>
```



En la consola verificamos el valor de la variable elementoli (nuevo elemento en lista)





## EJEMPLO

Vamos a crear un nodo de texto al cual se le asigna atributos y estilos por DOM para luego agregarlo a un elemento <div>

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>EJEMPLO 4 DOM</title>
</head>
<body>
<h1 id="titulo">Procesos de edición DOM</h1>

<p>DOM es una abreviatura de Document Object Model.
  En español podríamos traducirlo por Modelo de Objeto de Documento.
</p>

<p id="segundo">A través del DOM se puede acceder, por medio de Javascript, a cualquiera
de estos elementos,
  es decir a sus correspondientes objetos para alterar sus propiedades o invocar a sus
métodos.
</p>

<p>También, a través del DOM, queda disponible para los programadores de Javascript,
cualquier elemento de la página,
  para modificarlos, suprimirlos, crear nuevos elementos y colocarlos en la página, etc.
</p>
<div id="sub"></div>
<script>
//1. Asignación de variables de elementos con su Id y nombre de tag
var elementosP=document.getElementsByTagName("p");
var segundoP=document.getElementById("segundo");

//2. Crear elemento h2 (etiqueta)
```

```
var elementoH=document.createElement("h2");

//3. Crear un nodo de texto
var contexto=document.createTextNode("Este es la cadena de texto agregada en etiqueta h2");

//4. Añadir el nodo de texto a elemento
elementoH.appendChild(contexto);

//5. Agregar atributos y estilos al elemento nuevo
elementoH.setAttribute("align","center");
elementoH.style.color ="blue";

//6. Agregar el elemento al documento HTML
document.getElementById("sub").appendChild(elementoH);
</script>
</body>
</html>
```

## RESULTADO FINAL

### Procesos de edición DOM

DOM es una abreviatura de Document Object Model. En español podríamos traducirlo por Modelo de Objeto de Documento.

A través del DOM se puede acceder, por medio de Javascript, a cualquiera de estos elementos, es decir a sus correspondientes objetos para alterar sus propiedades o invocar a sus métodos.

También, a través del DOM, queda disponible para los programadores de Javascript, cualquier elemento de la página, para modificarlos, suprimirlos, crear nuevos elementos y colocarlos en la página, etc.

**Este es la cadena de texto agregada en etiqueta h2**

### 2.1.8 Como incluir JavaScript en documentos HTML

Para realizar esta operación el código se debe encerrar entre las etiquetas <script> y </script> y se puede agregar en cualquier parte del documento, aunque se recomienda incluirlo dentro de la etiqueta de encabezado de la página <head>; más adelante se dan algunos consejos adicionales al respecto según sea el caso.

Miremos un pequeño ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="description"/>
  <meta name="keywords" content="HTML5, CSS3, JavaScript" />
  <title> Ejemplo Uno JavaScript</title>
  <link rel="stylesheet" href="Estilos.css">

  <script type="text/javascript">
    alert("Este es un mensaje de prueba")
  </script>
</head>
<body>
  <h1>Encabezado de nivel 1</h1>
  <h2>Encabezado de nivel 2</h2>
  <h3>Encabezado de nivel 3</h3>
  <h4>Encabezado de nivel 4</h4>
  <h5>Encabezado de nivel 5</h5>
  <h6>Encabezado de nivel 6</h6>

  <p> Aquí estamos colocando elementos ya conocidos por
    el estudiante en el módulo anterior </p>
</body>
</html>
```

En el ejemplo anterior podemos observar encerrado entre un rectángulo de color rojo el script embebido en el documento HTML. Esta forma de trabajar con JavaScript tiene la desventaja de si se presenta la necesidad de realizar cambios en el código, sería necesario realizar la modificación en todas las páginas que dependan del mismo bloque de código. Pero tiene como ventaja que lo podemos utilizar cuando se requiere generar procesos específicos en una página en particular y cuando manejamos pequeños bloques de código.

### 2.1.9 JavaScript definido en un documento externo

El código de JavaScript se puede incluir en un archivo aparte, teniendo en cuenta que se debe guardar con la extensión js; a este archivo las páginas HTML se enlazan mediante las etiquetas `<script>` y `</script>` y podemos crear todos los archivos JavaScript que se requieran si es el caso.

Veamos un ejemplo:

El código JavaScript será:

```
alert ("este es un ejemplo con un archivo independiente")
```

Este script se debe guardar con la extensión js. Y la forma de enlazar la página HTML con el archivo anterior es:

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="description"/>
  <meta name="keywords" content="HTML5, CSS3, JavaScript" />
  <title> Ejemplo Uno JavaScript</title>
  <link rel="stylesheet" href="Estilos.css">

  <script type="text/javascript"
    src="D:\INFORMACION\informacion\artandres\Modulos\A2018\Js\PrimerScript.js"> </script>
</head>
<body>
  <h1>Encabezado de nivel 1</h1>
  <h2>Encabezado de nivel 2</h2>

  <p> Aquí vemos otro ejemplo </p>

</body>
</html>

```

Podemos observar que adicionalmente al atributo type se debe utilizar de igual manera el atributo src, el cual indica la ruta en donde se encuentra el archivo .js o JavaScript creado y con el cual se enlaza la página HTML; cabe anotar que cada etiqueta <script> enlaza un único archivo de JavaScript, pero en una página Web se pueden utilizar un número indeterminado de estas etiquetas si es el caso.

La mayor ventaja de utilizar este método es respecto al mantenimiento ya que minimiza o disminuye el código que se debe utilizar en la página HTML y el código JavaScript lo podemos reutilizar en otras páginas dependiendo de nuestras necesidades.

#### 2.1.10 Incluir JavaScript en los elementos HTML

Esta forma de utilizar JavaScript consiste en incluir partes de código JavaScript dentro del código HTML. La dificultad más grande que representa este método es que por decirlo de alguna manera viciamos el código HTML y se complica el mantenimiento del código escrito en JavaScript; de tal manera que este método no se recomienda y en nuestro caso no lo utilizaremos.

Veamos un ejemplo:

```

<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="description"/>
  <meta name="keywords" content="HTML5, CSS3, JavaScript" />
  <title> Ejemplo tres JavaScript</title>
  <link rel="stylesheet" href="Estilos.css">
</head>
<body>
  <h1>Encabezado de nivel 1</h1>
  <h2>Encabezado de nivel 2</h2>

  <p onclick="alert('Este mensaje se ejecuta de una manera diferente')" >
    Aquí vemos otro ejemplo.
  </p>

</body>
</html>

```

```

<body>
  <h1>Encabezado de nivel 1</h1>
  <h2>Encabezado de nivel 2</h2>

  <noscript>
    <p>
      Bienvenido a mi página WEB.
    </p>
    <p>
      Esta página requiere para su buen funcionamiento
      el uso de JavaScript. Se recomienda activarlo.
    </p>
  </noscript>

</body>
</html>

```

Para terminar si por alguna circunstancia el usuario de una página WEB deshabilita el uso de

JavaScript es razonable que podamos advertir de alguna manera que la pagina requiere de su uso; esto se hace utilizando la etiqueta <noscript> de la siguiente manera:

## FDEH

- El estudiante creara un script que permita mostrar sus datos personales y una bienvenida a sus compañeros de clase.
- El estudiante creara un script utilizando arrays que permita almacenar los meses del año y luego muestre el mes de febrero.
- El estudiante creara un script que permita almacenar en un arreglo la siguiente información: [5, true, false, 3, "Hola Amigos", Hola INCAP"] y con ella realizar las siguientes operaciones:
  - Determinar cuál de los elementos de texto es mayor, solo utilizando lo aprendido hasta ahora.
  - Utilizando solo los valores booleanos determinar las operaciones necesarias para que el resultado sea true.
  - Realizar las cuatro operaciones básicas matemáticas con los valores numéricos.

## UNIDAD 2

### METODOS, EVENTOS Y PROPIEDADES DE JAVASCRIPT

Utilizar de manera técnica dentro de la programación de una interfaz de usuario, los métodos, Condicionales, arrays, Bucles y funciones en JavaScript y generar de esta forma la interfaz gráfica de usuario adecuada.

#### FDH

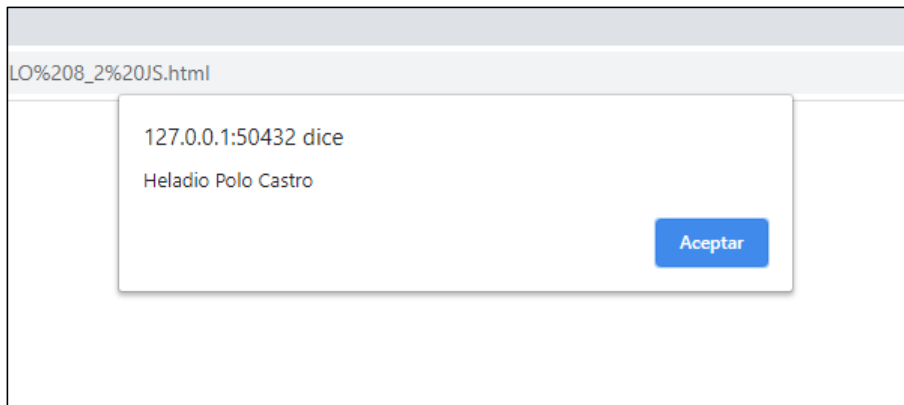
#### 2.2 Propiedad innerHTML

Esta propiedad se utiliza para almacenar el contenido de un elemento HTML, por lo general se utiliza un `document.getElementById` para posteriormente mediante la instrucción `alert` o `console.log` mostrar el contenido almacenado en el navegador web.

#### Ejemplo 1:

```
<html>
<head>
<meta charset="UTF-8">
</head>
<body>
<p id="texto"> </p>
<!-- Se crea la variable conten para asignarle el nodo de texto al elemento <p> con
id="texto" para después mostrarlo por medio de un cuadro de dialogo -->
<script type="text/javascript">
    var conten= document.getElementById("texto").innerHTML= "Heladio Polo Castro";
    alert(conten);
    document.write(conten);
</script>
</body>
</html>
```

Resultado final



## 2.3 Métodos String de Javascript

Estos métodos se utilizan para gestionar cadenas de texto.

### 2.3.1 Método length

Se utiliza para generar el largo de una cadena de texto.

<!-- Vamos a generar la cantidad de caracteres que tiene la cadena de texto nombre y apellidos, muestra 19 - ->

```
<script type="text/javascript">
```

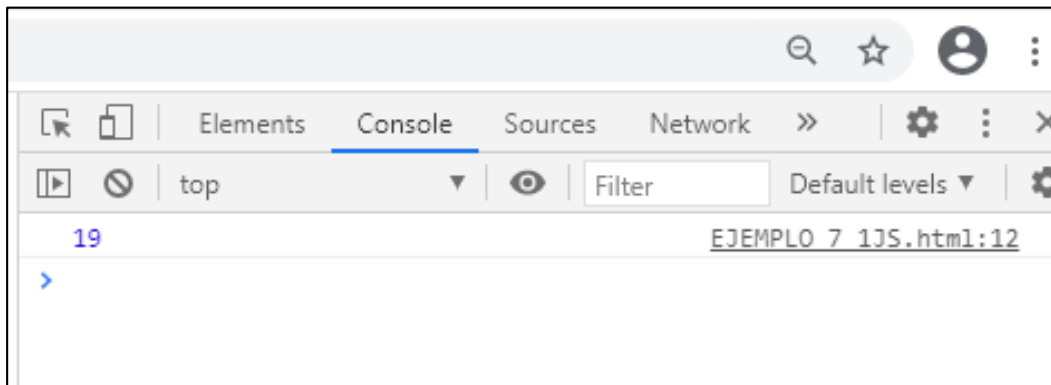
```
var b="Heladio Polo Castro";
```

```
var largo=b.length;
```

```
console.log(largo);
```

```
</script>
```

### Resultado final



### 2.3.2 Método concat()



Se utiliza para unir dos o más strings, se puede utilizar el operador + junto con las comillas simples ' ' o comillas dobles " " para unir dos valores.

### Ejemplo

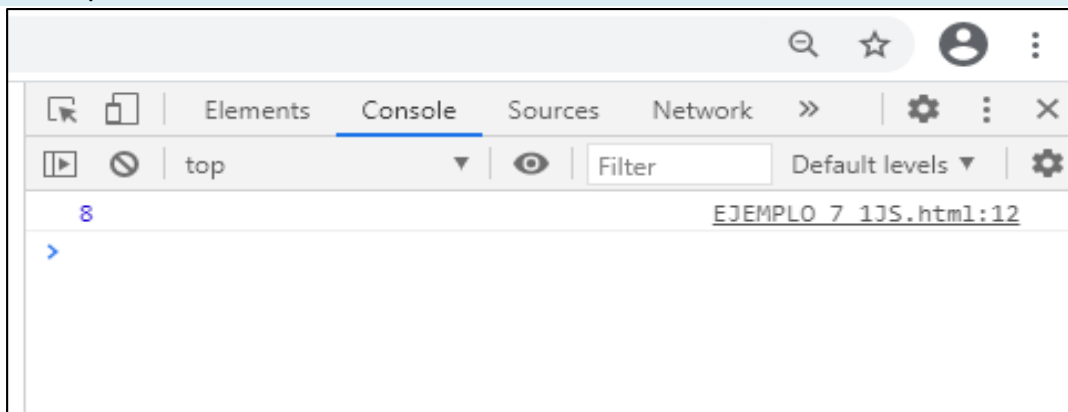
```
document.write(nombre + ' tu materia es aprobada con un...' + nota);  
document.write("nombre estudiante:"+" "+nombre);
```

### 2.3.3 Método Indexof()

Devuelve el índice de la posición de la primera letra de la cadena de texto a buscar, aclarando que la búsqueda comienza de izquierda a derecha a partir del índice 0, luego 1,2,3,...n; además se utiliza para búsquedas en otras estructuras como arrays y tablas.

// Vamos a generar la posición donde encuentra el apellido Polo en la cadena de texto, muestra 8

```
<script type="text/javascript">  
var b="Heladio Polo Castro";  
var largo=b.indexOf("Polo");  
console.log(largo);  
</script>
```

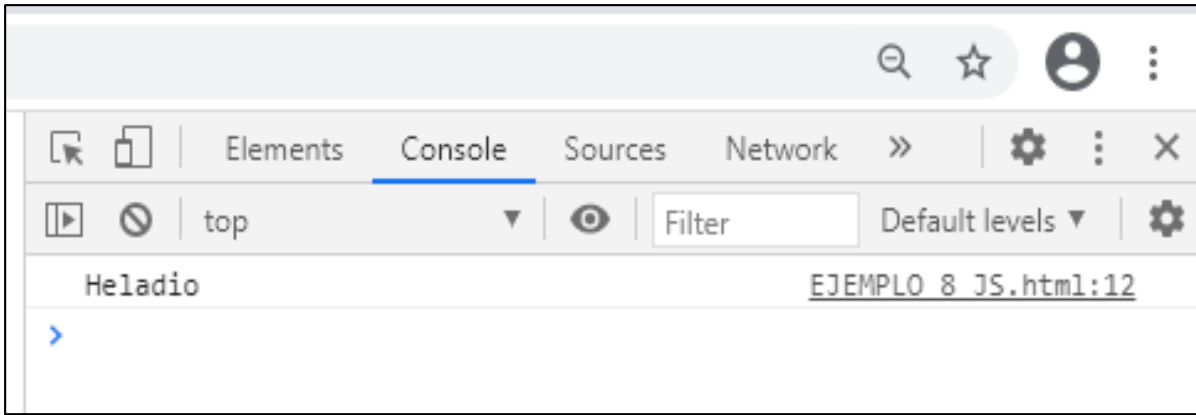


### 2.3.4 Método Substring()

Permite hallar de acuerdo a la posición inicial y la posición final la cadena de texto resultante.

// Vamos a generar la cadena de texto de acuerdo a su ubicación dando la posición Inicial y la posición final, muestra Heladio

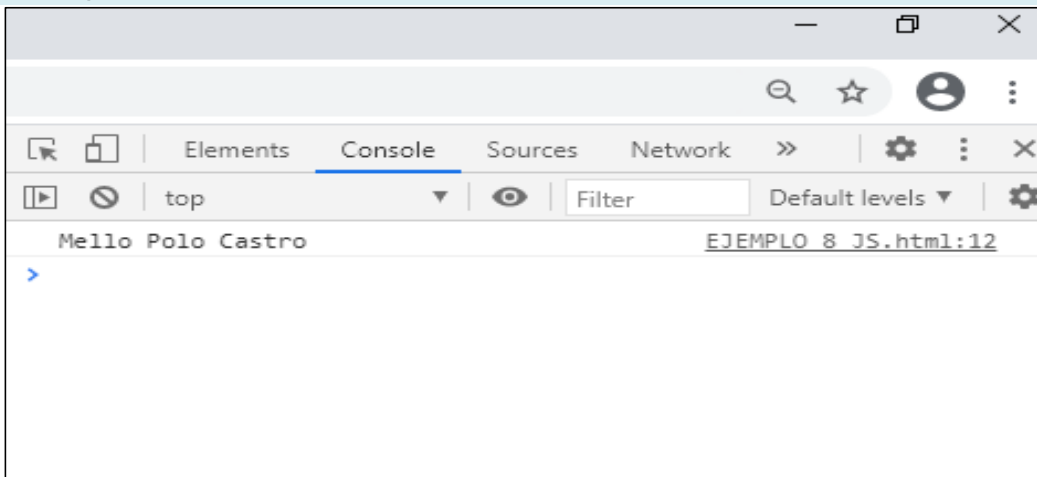
```
<script type="text/javascript">  
var b="Heladio Polo Castro";  
var largo=b.substring(0,7);  
console.log(largo);  
</script>
```



### 2.3.5 Método Replace()

Permite sustituir un nuevo texto dentro de una cadena de texto previa localización.

```
<script type="text/javascript">  
// Vamos a reemplazar el texto Heladio por el texto Mello, mostrará Mello Polo Castro  
var b="Heladio Polo Castro";  
var largo=b.replace("Heladio","Mello");  
console.log(largo);  
</script>
```



### 2.3.6 Función toUpperCase

Transforma los caracteres en mayúscula. Ejemplo:

```

<script>
    var mensaje1 = "Hola Compañeros";
    var mensaje2 = " INCAP";
    var mensaje = mensaje1 + mensaje2;

    alert(mensaje);

    var mensaje3 = "hola";
    var mensajeconcatenar = mensaje3.concat(" incap");
    var mensajemayusculas=mensajeconcatenar.toUpperCase();

    alert(mensajemayusculas);

</script>

```

### 2.3.7 Función toLowerCase

Transforma o convierte los caracteres en minúscula si se encuentran en mayúscula.

Ejemplo:

```

<script>
    var mensaje1 = "Hola Compañeros";
    var mensaje2 = " INCAP";
    var mensaje = mensaje1 + mensaje2;

    alert(mensaje);

    var mensaje3 = "HOLA";
    var mensajeconcatenar = mensaje3.concat(" INCAP");
    var mensajeminusculas=mensajeconcatenar.toLowerCase();

    alert(mensajeminusculas);

</script>

```

A pesar de que existen muchas más funciones para manejo de cadena de caracteres tratamos solo las anteriores ya que son las más utilizadas; las demás quedaran a discreción del formador para estudiarlas y aplicarlas.

## 2.4 Métodos Numéricos de Javascript

Estos métodos se utilizan para gestionar valores numéricos.

### 2.4.1 Método toString()

Se utiliza para convertir un numero en una cadena de texto.

Ejemplo:

// En este ejemplo el valor numérico 157 se convierte en un texto

```
<script type="text/javascript">
var numx = 157;
var num1= numx.toString();
alert("toString-->"+num1);
</script>
```

127.0.0.1:50432 dice

toString--> 157

Aceptar

### 2.4.2 Método toFixed()

Se utiliza para generar un numero con decimales dependiendo de los números decimales solicitados en expresión si es 0 se aproxima al número más alto siguiente.

Ejemplo:

```
<script type="text/javascript">
// En este ejemplo el valor numérico 15.757 utilizando el método toFixed y el parámetro 0 lo
aproxima al número siguiente más alto =16
var numx = 15.757;
var num1= numx.toFixed(0);
var num2= numx.toFixed(2);
alert("toFixed(0)-->"+num1+" "+ "\n toFixed(2)--->"+num2);
</script>
```

127.0.0.1:50432 dice

toFixed(0)--> 16

toFixed(2)---> 15.76

Aceptar

### 2.4.3 Método parseInt

Se utiliza para convertir en un número entero de una cadena de texto.

```
<h1>Metodo parseInt</h1>
```

```
<p id="texto"></p>
```

```
<script type="text/javascript">
```

```
// En este ejemplo el método parseInt convierte la cadena de texto 15 años a un número entero
```

```
var numx = "15 años";
```

```
var num1= parseInt(numx);
```

```
//
```

```
document.getElementById("texto").innerHTML= num1;
```

```
alert("parseInt-->" + num1);
```

```
</script>
```

127.0.0.1:50432 dice

parseInt--> 15

Aceptar

### 2.4.4 Método isNaN

Devuelve falso si un número es correcto o verdadero si no es un número o es una expresión matemática incorrecta.

Ejemplo:

```
<p id="texto"></p>
```

```

<script type="text/javascript">
// En este ejemplo el método isNaN verifica si el dato 15 años no es un número y su
resultado es verdadero. (porque es una cadena de texto)
var numx = "15 años";
  var num1= isNaN(numx);
  //
  document.getElementById("texto").innerHTML= num1;
  alert("isNaN-->" + num1);
</script>

```

127.0.0.1:50432 dice

isNaN-->true

Aceptar

### 2.4.5 Método parseFloat

Se utiliza para convertir en un numero con decimales de una cadena de texto.

```

<h1>Metodo parseFloat</h1>
<p id="decimal"></p>
<script type="text/javascript">
// En este ejemplo el método parseFloat convierte un número ingresado a un número con
decimales combinado con el método toFixed.
var numx = 4.59678;
  var num1= parseFloat(numx);
  document.getElementById("decimal").innerHTML= num1.toFixed(2);
  alert("parseFloat-->" + num1.toFixed(2));
</script>

```



## 2.5 Métodos Arrays de Javascript

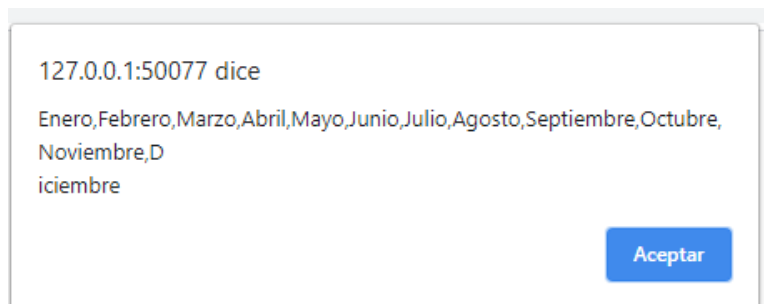
### 2.5.1 Método toString()

Este se utiliza para convertir los valores integrantes de un arreglo vector en un array de strings.

Ejemplo

```
<h1 id="cabecera">METODO ARRAYS EN JAVASCRIPT</h1>
<p id="texto"></p>

<script type="text/javascript">
// En este ejemplo el método toString( ) se utiliza para convertir los elementos de un Arrays
en un arreglo de textos.
var mes_list
=['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviemb
re','Diciembre'];
document.getElementById("texto").innerHTML=mes_list.toString();
var lista = mes_list.toString();
alert(lista);
</script>
```



### 2.5.2 Método POP()

Método que se utiliza eliminar el último componente de un arreglo tipo vector.

Ejemplo

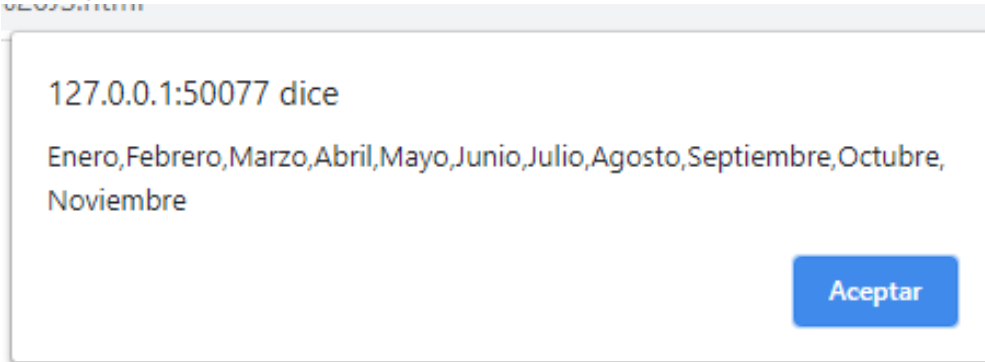
```
<h1 id="cabecera">METODO POP EN JAVASCRIPT</h1>
<p id="texto"></p>
<script type="text/javascript">
// En este ejemplo el método pop( ) se utiliza para eliminar el último elemento (Diciembre) de
un array.
```

```

var mes_list
=['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviemb  

e','Diciembre'];
document.getElementById("texto").innerHTML=mes_list.pop();
var lista = mes_list;
alert(lista);
</script>

```



### 2.5.3 Método PUSH()

Método que se utiliza agregar un componente a un arreglo tipo vector (al final).

#### Ejemplo

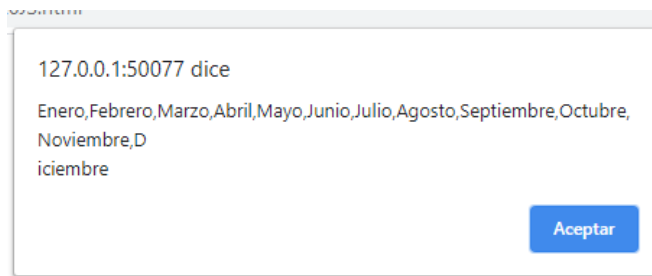
```

<h1 id="cabecera">METODO PUSH EN JAVASCRIPT</h1>
<p id="texto"></p>
<script type="text/javascript">
// En este ejemplo el método push( ) se utiliza para agregar el elemento (Diciembre) al final
de un array.
var mes_list
=['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviemb  

e'];
document.getElementById("texto").innerHTML=mes_list.push('Diciembre');
var lista = mes_list;
//
alert(lista);
</script>

```



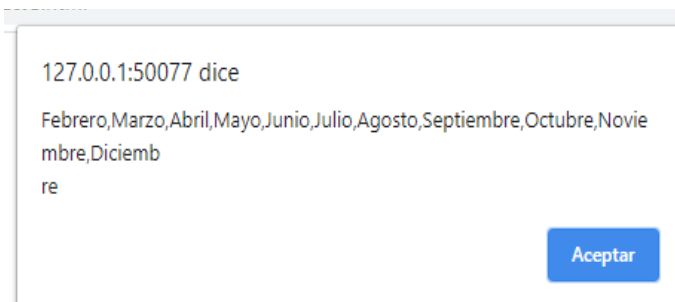


### 2.5.4 Método SHIFT()

Método que se utiliza para eliminar un componente a un arreglo tipo vector (al principio), a su vez nos muestra el componente del array que se suprimió.

#### Ejemplo

```
<h1 id="cabecera">METODO SHIFT EN JAVASCRIPT</h1>
<p id="texto"></p>
// En este ejemplo el método shift( ) se utiliza para eliminar del array el elemento (Enero) al
// principio del mismo.
<script type="text/javascript">
var mes_list
=['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];
document.getElementById("texto").innerHTML=mes_list.shift();
var lista_nueva =mes_list
//
alert(lista_nueva);
</script>
```

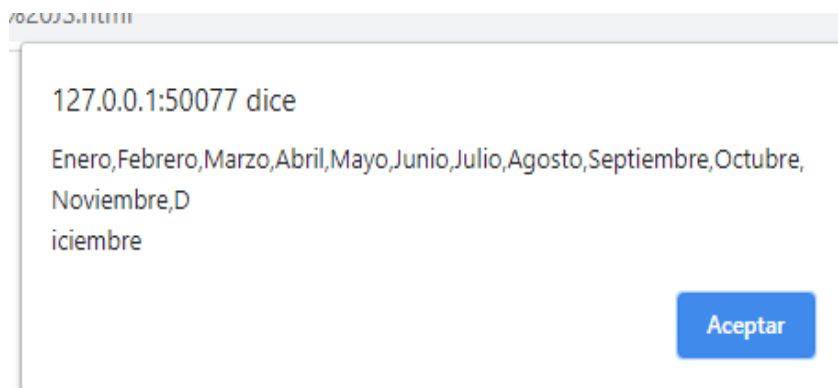


### 2.5.5 Método UNSHIFT()

Método que se utiliza para agregar un componente a un arreglo tipo vector (al principio), a su vez nos da la longitud del nuevo array.

Ejemplo:

```
<p id="texto"></p>
//En este ejemplo el método unshift( ) se utiliza para agregar en array el elemento (Enero)
al principio del mismo.
<script type="text/javascript">
var mes_list
=['Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre','Octubre','Noviembre','Diciembre'];
document.getElementById("texto").innerHTML=mes_list.unshift('Enero');
var lista_nueva =mes_list
//
alert(lista_nueva);
</script>
```



### 2.5.6 Método SPLICE()

Se utiliza para eliminar un elemento del array en un índice deseado

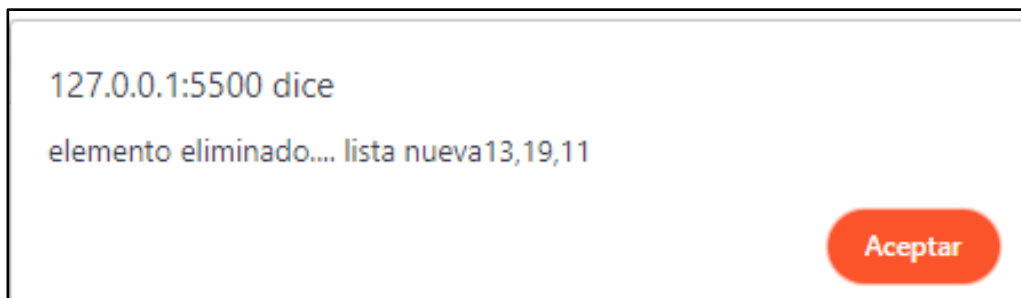
Ejemplo:

```

<h1>Metodo splice</h1>
<p id="decimal"></p>
<script type="text/javascript">
    var lista = [13, 15, 19, 11];
    // primero se busca número 15 con metodo indexOf, en array nos da la posición del
    elemento a eliminar
    var indice = lista.indexOf(15);
    // luego se elimina el elemento encontrado en array usando splice, recuerda que todo
    indice de un array comienza por 0:
    if (indice > -1) {
        var index = lista.splice(indice, 1);
        alert("elemento eliminado.... lista nueva" + lista);
    }
    else
    {
        alert("no se encontró elemento" );
    }
</script>

```

## Resultado final



### 2.5.7 Método MATH

Se utilizan para realizar operaciones matemáticas con valores numéricos.

**Math.round ()** ...Redondea número al entero más próximo (a partir de 0.5 lo aproxima hacia el entero mayor, si es menor a 0.5 lo aproxima al entero menor).

**Math.abs ()** ... Genera el valor absoluto o positivo del número.

**Math.floor ()** ...Redondea número al entero más próximo (por debajo).

**Math.ceil ()** ...Redondea número al entero más próximo (por arriba).

**Math.random ()** ...Genera un numero aleatorio de 0 a 1.

### Ejemplo

**Math.round(3.67);** .... //resultado 4

**Math.round(3.47);** .... //resultado 3

**Math.abs(-4.5);** .... //resultado 4.5

### FDEH (Formador dice y estudiante hace)

✓ Estudiante para afianzar conocimientos adquiridos debes de colocar el método correcto  
var sem\_list = ['lunes','martes','Miercoles','jueves','viernes','sabado'];  
document.getElementById("texto").innerHTML=sem\_list.\_\_\_\_('Domingo');

✓ Utilice HTML DOM para cambiar el valor del campo de entrada.

<input type="text" id="dato" value="Dar clic">

<script>

document.getElementById("dato").\_\_\_\_\_ = "Javascript";

</script>

## 2.6 Estructuras secuenciales de un programa

### 2.6.1 Captura de datos en Javascript

**Instrucción prompt()** : este comando me permite generar un cuadro o ventana de dialogo para permitir el ingreso de información por teclado, por lo general se asigna a una variable determinada; su sintaxis es:

**Var nombrevar =prompt('Mensaje de entrada','Valor por defecto');**

Ejemplo:

// En este ejemplo se declara las tres variables y luego se pide por teclado (prompt) los datos de nombre, precio y cantidad de un producto.

var precio;

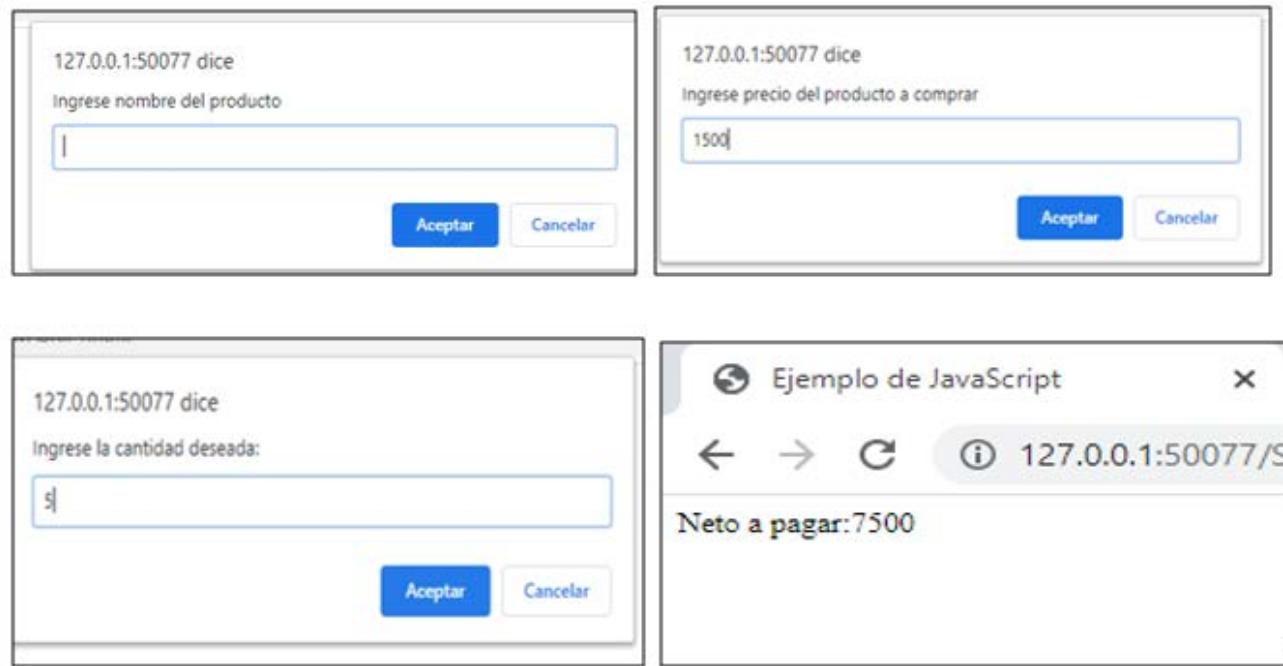
var cant;

var nombrep;

nombrep=**prompt**('Ingrese nombre del producto,');

precio=**prompt**('Ingrese precio del producto a comprar,');

cant=**prompt**('Ingrese la cantidad deseada:');



El comando anterior viene acompañado cuando se quiere que el resultado se muestre directamente en la página HTML; para ello utilizaremos el comando *document.write()* El cual es una de las salidas donde se escribe en el navegador el texto que se le indique.

## 2.6.2 Salida de datos en Javascript

### Instrucción `document.write()`;

Este comando nos permite mostrar los resultados de los procesos generados en la salida del navegador. Veamos un ejemplo que muestra las ventajas de dichos comandos, su sintaxis es:

**`Document.write('mensaje' + variable)`**

// En este ejemplo se declara las tres variables y luego se pide por teclado (prompt) los datos de nombre, precio y cantidad de un producto y al final se muestra en la salida del navegador, la instrucción `document.write` del valor a pagar.

```
<script>
var precio;
var cant;
var nombrep;
```

```
nombrep=prompt('Ingrese nombre del producto,');  
precio=prompt('Ingrese precio del producto a comprar,');  
cant=prompt('Ingrese la cantidad deseada:', '');  
var netop;  
netop=parseInt(precio)*parseInt(cantidad);  
document.write('Neto a pagar:');  
document.write(netop);  
</script>
```

### 2.6.3 Ejemplos con instrucciones básicas

#### EJEMPLO 1

En los siguientes ejemplos utilizaremos variables diversas y la instrucción `document.write`

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>EJEMPLO 1 JAVASCRIPT</title>  
    <meta charset="UTF-8">  
</head>  
<body bgcolor=lightblue>  
<!-- Se crean y asignan valores a las variables de tipo texto, numéricas y booleanas al final se  
muestran en la salida del navegador. -->  
  
    <script type="text/javascript">  
        var nombre='Pedro';  
        var edad=50;  
        var altura=1.75;  
        var soltero=false;  
        document.write(nombre);  
        document.write('<br>');  
        document.write(edad);  
        document.write('<br>');  
        document.write(altura);  
        document.write('<br>');
```

```
        document.write(soltero);  
    </script>  
</body>  
</html>
```

## EJEMPLO 2

Desarrolle un script para generar el área de un triángulo.

```
<!DOCTYPE html>  
<html>  
<head>  
    <title>Ejemplo 2 de instrucciones de JavaScript</title>  
    <meta charset="UTF-8">  
</head>  
<body>  
    <!--Se crean y asignan valores a las variables numéricas base y altura para generar el área del  
    triángulo al final se muestran en la salida del navegador. -->  
  
    <script>  
        var base;  
        var altura;  
        base=prompt('Ingrese la base del triangulo:','');  
        altura=prompt('Ingrese altura del triangulo');  
        var areat;  
        areat=(base*altura)/2;  
        document.write('El area del triangulo es:');  
        document.write(areat);  
    </script>  
</body>  
</html>
```

## EJEMPLO 3

Desarrolle un script que permita ingresar 4 valores distintos y genere una sumatoria y una

productoria con esos números y muestre en pantalla los resultados respectivos.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo 3 de instrucciones de JavaScript </title>
  <meta charset="UTF-8">
</head>
<body>
<!-- Se crean y asignan cuatro valores a cuatro variables numéricas se pide por teclado y se
genera la sumatoria y la productoria de los números al final se muestran en la salida del
navegador. -->
<script type="text/javascript">
  var num1;
  var num2;
  var num3;
  var num4;
  num1=prompt('Ingrese primer valor:','');
  num2=prompt('Ingrese segundo valor:','');
  num3=prompt('Ingrese tercer valor:','');
  num4=prompt('Ingrese cuarto valor:','');
  var suma;
  suma=parseInt(num1)+parseInt(num2)+parseInt(num3)+parseInt(num4);
  var producto;
  producto=parseInt(num1)*parseInt(num2)*parseInt(num3)*parseInt(num4);
  document.write('La suma de los cuatro valores es:');
  document.write(suma);
  document.write('<br>');
  document.write('El producto de los cuatro valor es:');
  document.write(producto);
</script>
</body>
</html>
```

#### EJEMPLO 4

Desarrollar un script donde se pida por teclado los siguientes datos: nombre, precio y



cantidad de un producto a facturar y mostrar por pantalla su valor antes de iva, su descuento 10%, su iva 16% y su neto a pagar.

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>
<script>
  var precio, cant, desc, iva, subtot, netop;
  var nombrep;
  // Se crean y asignan siete valores a siete variables se piden por teclado y se genera los
  // datos de una factura subtotal, descuento, iva y el neto a pagar al final se muestran los
  // resultados en la salida del navegador..
  nombrep=prompt('Ingrese nombre del producto,');
  precio=prompt('Ingrese precio del producto a comprar,');
  cant=prompt('Ingrese la cantidad deseada,');
  //
  subtot=parseInt(precio)*parseInt(cant);
  desc=subtot*0.10;
  iva=(subtot-desc)*0.16;
  netop=subtot-desc+iva;
  //
  document.write('SUBTOTAL:'+'<br>');
  document.write(subtot+'<br>');
  document.write('Descuento:'+'<br>');
  document.write(desc+'<br>');
  document.write('Iva:'+'<br>');
  document.write(iva+'<br>');
  document.write('Neto a pagar:'+'<br>');
  document.write(netop);
</script>
</body>
</html>
```

## **FDEH (formador dice y estudiante hace)**

Estudiante para afianzar este conocimiento debes de realizar los siguientes scripts:

- ✓ Sumar tres números que da el usuario
- ✓ Calcular el promedio de un estudiante el cual ingresa 3 notas
- ✓ Hallar el producto de 3 números y mostrar el resultado obtenido.
- ✓ Hallar el valor del día de un empleado que gana \$8.000 por hora.
- ✓ Diseñe un scripts para determinar el área del cuadrado.
- ✓ Diseñe un scripts para determinar el área de un rectángulo.
- ✓ Diseñe un scripts que me permita calcular el porcentaje de hombres y de mujeres de un número cualquiera de personas (se debe pedir el número de personas, el número de hombres y el número de mujeres), enviar el porcentaje de hombres y de mujeres por pantalla.
- ✓ Diseñe un scripts que permita generar la velocidad de un vehículo teniendo en cuenta su espacio recorrido y su tiempo gastado.

FDH

## **2.7 Estructuras de control de flujo de un programa**

### **2.7.1 Transferencia de control condicional básica(if)**

Esta estructura de programación es muy en los diferentes lenguajes de programación ya que permite que un programa evalúe una o varias condiciones y dependiendo del resultado de dicha evaluación el propio programa tome decisiones si el resultado de la misma es falso (false) o es verdadero (true).

Su sintaxis en JavaScript es:

```
if (condición) {  
Procesos por verdadero;
```

```

}
Else {
Procesos por falso;
}

```

Si la condición analizada o evaluada se cumple o su resultado es true entonces se procede a realizar el o los procesos por verdadero que se encuentran dentro de las llaves {...}; si no se cumple se procede a realizar los procesos por falso que están limitados por las llaves después el programa sigue su flujo normal.

Veamos un ejemplo donde aplicamos el condicional básico:

```

<script>
var nombre;
var nota;
nombre=prompt('Ingrese nombre estudiante:', '');
nota=parseFloat(prompt('Ingrese su nota final:', ''));
if (nota >= 3)
{
document.write(nombre + ' tu materia es aprobada con un ' + nota);
}
else
{
document.write(nombre + ' tu materia es reprobada con un ' + nota);
}
</script>

```

## FDEH

✓ Complete y corrija la instrucción if para enviar una alert "Estudia Javascript" si x es mayor que y, de lo contrario, enviar un alert "Estudia PHP".

```

If ( _____){
_____
}_____ {
_____
}

```

✓ Complete la instrucción if que ingresa dos valores al azar y se debe de realizar las comparaciones pertinentes.

```
if ( _____ ) {  
alert("El Valor1 No es mayor que el Valor2");  
}
```

```
if ( _____ ) {  
alert( "El Valor2 es positivo")  
}
```

```
if ( _____ ) {  
alert("El Valor1 es negativo o diferente de cero")  
}
```

**FDH**

### 2.7.2 Transferencia de control condicional anidado (if/else)

En la vida real y en los procesos empresariales reales, lo normal es que se presente una condición y dependiendo del resultado de la evaluación de la misma se deban realizar varios procesos si el resultado es verdadero (true) o Y procesos si el resultado es falso (false); es entonces, cuando la condicional con esta estructura (if/else) se convierte en una herramienta de vital importancia.

Su sintaxis general es:

```
if (condición) {  
Procesos por verdadero; }  
Else if (condición) {  
    Procesos por verdadero;  
}  
Else {  
Procesos por falso;  
}
```

La sintaxis indica que si la condición evaluada es verdadera, se realizarán los procesos que están dentro del bloque `if()`; de lo contrario, ósea si la condición no se cumple o es falsa se realizarán los procesos que se encuentran dentro del bloque `else {}` y se evalúa la siguiente condición ya en la última se evalúa la condición falsa o última comparación.

Veamos un ejemplo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo de JavaScript</title>
  <meta charset="UTF-8">
</head>
<body>
<p> INGRESE NOTAS DE 1 A 5</p>
<script>
  var nota1,nota2,nota3,prom;
  nota1=parseFloat(prompt('Ingrese nota n1:',''));
  nota2=parseFloat(prompt('Ingrese nota n2:',''));
  nota3=parseFloat(prompt('Ingrese nota n3:',''));
  prom=(nota1+nota2+nota3)/3;
  //
  if (prom<=2.9)
  {
    document.write('Reprobado...' + prom);
  }
  else if (prom<=3.9) {
    document.write('Regular...' + prom);
  }
  else if (prom<=5.0) {
    document.write('Sobresaliente...' + prom);
  }
  Else {
    document.write('Sobresaliente...');
```

```
    }  
</script>  
</body>  
</html>
```

### 2.7.3 Ciclos repetitivos o Bucles:

#### 2.7.3.1 Ciclo for():

Cuando dentro de un programa de computador se requiere que uno o varios procesos se repitan varias veces, entonces es necesario recurrir a los ciclos repetitivos o bucles y déjeme decirle que esta situación se presenta muchas pero muchas veces en la vida real; Uno de los ciclos repetitivos que más se utiliza de pronto por su fácil comprensión es el for(), y sobre todo en procesos con arreglos o arrays. La sintaxis general es:

```
for (inicialización; Condición; incremento) {  
  Procesos a repetir;  
}
```

Explicaremos brevemente cada uno de los argumentos encerrados entre paréntesis:

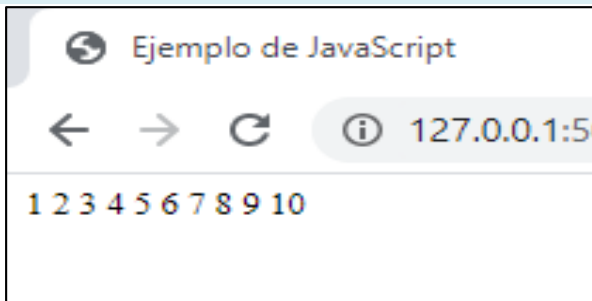
- Inicialización: este argumento establece el punto de partida el valor inicial de la variable contador, encargada de controlar las veces que el ciclo se debe repetir.
- Condición: es la condición o criterio que determina si se continua con el ciclo o se termina el mismo.
- Incremento: es la forma en la que se controla que la variable contadora cambie y se incremente para que dicho ciclo tenga un final; de lo contrario y si esto no ocurre crearíamos un ciclo infinito u también llamado un loop repetitivo.

Veamos un ejemplo:

```
<!-- En este ejemplo creamos un ciclo con la instrucción for para generar los números  
consecutivos desde del 1 al 10. Se inicializa la variable f con 1 y finaliza hasta el 10 con un  
incremento de 1 en 1. -->
```

```
<script>  
  var f;  
  for(f=1;f<=10;f++)  
  {  
    document.write(f+" ");
```

```
}  
</script>
```

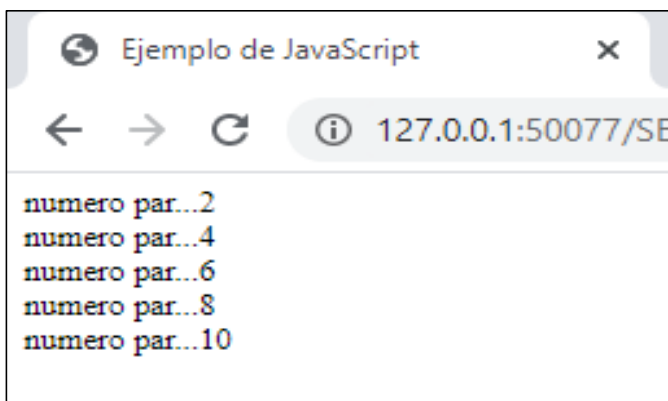


Miremos otro ejemplo generar los números pares del 1 al 10:

/<! - -En este ejemplo creamos un ciclo con la instrucción for unido a la instrucción if para generar los números pares desde del 2 al 10.

Recuerda: se utiliza el símbolo == para comparar con 2. - - >

```
<script>  
var f;  
for(f=1;f<=10;f++)  
{  
  if (f % 2 == 0){  
    document.write("numero par..." + f + "<br>");  
  }  
}  
</script>
```



### 2.7.3.2 Ciclos while

En estos tipos de ciclos se evalúa primero la condición si es verdadera se realizan las instrucciones seguidas en caso que la condición sea falsa sale del ciclo y continúa el flujo del programa con las instrucciones siguientes a las llaves.

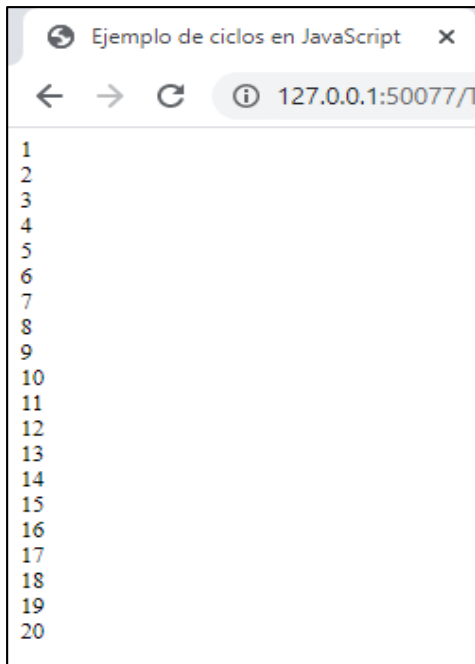
Ejemplo:

/<! - - En este ejemplo se generan los números consecutivos del 1 al 20. Se inicializa una variable num en 1 y se escribirá en el navegador web hasta el numero 20 mientras sea verdadero. - ->

```
<script>
  var num;
  num=1;
  while (num<=20)
  {
    document.write(num+"<br>");
    num++;
  }
</script>
```

### RESULTADO FINAL





### 2.7.3.3 Estructura switch:

Esta instrucción es aplicada al implementar menús de opciones, cuando varios resultados dependen de una sola condición u opción, en lugar de utilizar la sentencia if ( ) es recomendable utilizar la sentencia switch, cuya sintaxis es la siguiente:

```
Switch(variable){  
case valor1:  
Procesos valor 1;  
break;  
  
case valor2:  
Procesos valor 2;  
break;  
  
case valorN:  
Procesos valor N;  
break;  
  
default:
```

```
.....;  
break;  
}
```

## RESULTADO FINAL

```
<script>  
  
  var dia=parseInt(prompt('Digite un numero correspondiente al dia de la semana: '));  
  
  switch(dia) {  
    case 1:  
      alert ('Lunes');  
      break;  
    case 2:  
      alert ('Martes');  
      break;  
  
    case 3:  
      alert ('Miercoles');  
      break;  
    case 4:  
      alert ('Jueves');  
      break;  
    case 5:  
      alert ('Viernes');  
      break;  
    case 6:  
      alert ('Sabado');  
      break;  
    case 7:  
      alert ('Domingo');  
      break;  
    default:  
      alert ('numero invalido');  
      break;  
  }  
</script>
```

## FDEH (Formador dice y estudiante hace)

- El estudiante creara un script que permita calcular el neto a pagar en una nómina de cinco empleados, la cual está determinada por las siguientes condiciones:
- Al empleado se le paga un salario básico determinado por el contrato firmado.
- Al empleado se le hace un descuento en pensión y salud correspondiente al 8% del salario básico devengado.
- Al empleado se le cancela un auxilio de transporte según lo establecido por la ley si el salario de dicho empleado es menor o igual a dos salarios mínimos legales vigentes establecidos por la ley colombiana.
- Se debe mostrar por pantalla el nombre del empleado y su neto a pagar.
- Crear un script que me permita crear un arreglo para guardar los meses del año y que luego

los muestre uno a uno.

FDH (Formador dice y hace)

## 2.8 Programación avanzada

Hasta ahora los scripts que se han desarrollado son de manejo básico y no requieren una estructura de programación muy compleja; en la vida real las aplicaciones en donde se involucra JavaScript requieren de mayor complejidad y es cuando el tema que a continuación expondremos, adquiere una importancia sustancial para que las aplicaciones a desarrollar funcionen de manera adecuada y eficiente.

Antes de adentrarnos en el concepto y ejemplos de Funciones creadas por el programador o desarrollador, veamos algunos temas que harán nuestra experiencia de programación más amena y fácil.

### 2.8.1 Funciones Personalizadas

Cuando se requiere de ejecutar una instrucción o conjunto de instrucciones muchas veces dentro de una aplicación de mayor complejidad, es necesario recurrir a las funciones creadas por el programador o desarrollador para que de esta manera solo sea hacer un llamado a dicha función (invocarla) en el momento en que se desee utilizar.

Veamos la sintaxis de una función:

```
function nombre de la función([parámetros]){  
  Instrucciones de la función;  
}
```

**Function** es la palabra reservada que define la función.

Nombre de la función es el nombre que se dará a la función y con el cual se hará referencia al momento de ser llamada.

Parámetros son los valores de entrada que recibe la función para ser procesados; estos parámetros son opcionales y se deben colocar solo si la función los requiere.

Instrucciones de la función, son todos los procesos que debe realizar la función y van encerrados entre corchetes {}.

Para ejecutar la función tendremos que invocarla dentro de la página y en cualquier parte de esta; para estos procesos solo se debe escribir el nombre de la función seguido de los paréntesis.

NombreFuncion();

Veamos un ejemplo sencillo:

<! - - Se crea función bienvenida para que se pida por teclado el nombre del usuario, se asigna a una variable, luego llama a la función bienvenida y finalmente se ejecuta y envía mensaje de bienvenida pertinente por pantalla.- - >

```
<script type="text/javascript">
```

```
function bienvenida(nombre){
```

```
  alert( 'hola como estas...' + nombre);
```

```
}
```

```
var nombre=prompt( "digite su nombre");
```

```
bienvenida(nombre);
```

```
</script>
```

RESULTADO FINAL



Aquí tenemos otro ejemplo, pero embebido en el cuerpo <body> de la página HTML:

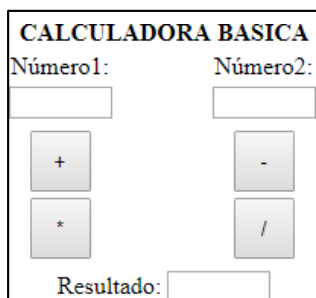
```
<script>
    function suma_y_muestra(primerNumero, segundoNumero) {
        var resultado = primerNumero + segundoNumero;
        alert("El resultado es " + resultado);
    }
    var numero1=3;
    var numero2 = 5;
    suma_y_muestra(numero1, numero2);
</script>
```

Vamos a listar algunas características de los argumentos de una función:

- ✓ El número de argumentos que se envían a una función, se recomienda que sea el mismo que se ha indicado en la función, pero esto no es obligatorio y JavaScript no muestra error si el número de argumentos es diferente.
- ✓ El orden de los argumentos es de vital importancia y tiene que ser estricto, tal cual se establece en la función.
- ✓ Se puede utilizar el número que se desee de parámetros, aunque se recomienda que no se exagere en el mismo.
- ✓ No es obligatorio que los nombres de los argumentos que se definen en la función sean los mismos nombres que se le pasan.

FDEH (Formador dice y estudiante hace)

- Escribir el código de una función a la que se pasa como parámetro un número entero y devuelve como resultado una cadena de texto que indica si el número es par o impar. Mostrar por pantalla el resultado devuelto por la función.
- Crear una función que me permita realizar las operaciones básicas en una calculadora sencilla como la que se muestra en la imagen siguiente:



FDH (Formador dice y hace)

## 2.9 Eventos en JavaScript

Evento	Descripción	Elementos para los cuales esta asociado o aplicado
onblur	Deseleccionar el elemento	Button, input, label, select, textarea, body
onchange	Cuando cambia un elemento	Input, select, textarea
onclick	Click sobre un elemento	Todos los elementos
onselect	Seleccionar texto en input	Input, textarea
onfocus	Seleccionar un elemento	Button, input, label, select, textarea, body
onkeydown	Pulsar una tecla sin soltar	Elementos del formulario y body

onkeypress	Pulsar una tecla	Elementos del formulario y body
onkeyup	Soltar una tecla pulsada	Elementos del formulario y body
onload	Página cargada totalmente	body
onmousedown	Pulsar sin soltar el mouse	Todos los elementos
onmousemove	Mover el ratón	Todos los elementos
onmouseover	Pasar por encima del elemento	Todos los elementos
onmouseout	Salir del elemento	Todos los elementos
onsubmit	Click en el botón enviar-datos	Form
onunload	Cerrar la pagina	body
onreset	Click en el botón cancelar-limpiar	Form

*Es de aclarar que los eventos más utilizados son **onload**, **onclick**, **onsubmit**, **onfocus**, **onreset**.*

Para que un evento surta efecto se debe asociar a un atributo de los elementos de HTML, a una función JavaScript externa o a un manejador semántico; este último no se estudiara en este módulo.

Veamos entonces como seria:

- Manejar un evento con un atributo del elemento: es la forma más sencilla pero la menos

recomendada ya que se debe incluir el código JavaScript en el atributo del elemento HTML. Miremos un ejemplo.

```
<input type="button" value="Da Click aqui" onclick="alert('gracias por dar clic en el botón');"/>
```

Si nos podemos dar cuenta el evento lo involucramos como atributo del elemento (botón de comando) y solo realiza la acción de enviar un mensaje de alerta cuando se produce dicho evento sobre el objeto. Imaginémonos que fueran más las acciones que se deberían desencadenar mediante este evento; sería menos fácil de manejar y "ensuciaría" en código HTML, aunque es una técnica que se puede utilizar, aunque es la menos recomendada.

Las funciones externas igualmente son un método muy sencillo pero que en aplicaciones complejas presentan alguno u otro problema; continuando con el ejemplo del apartado anterior, este mismo ejemplo utilizando una función externa se convertiría en lo siguiente:

```
function mostrarmensaje(){  
    alert("Gracias por dar clic en el boton");  
}  
<input type="button" value="Da clic aquí" onclick="mostrarmensaje()" />
```

En esta técnica hacemos el llamado a la función que realiza la acción desde el evento del elemento HTML, de tal manera que si la función realiza más instrucciones estas se están definiendo dentro de la función creada y no ensucia el código HTML.

Vamos a crear un ejemplo con el evento ONCLICK con botones que abren y cierran ventanas observemos:

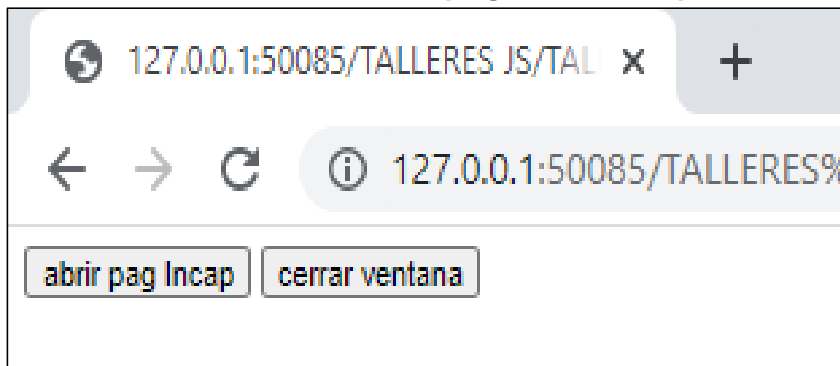
#### **Ejemplo:**

```
<script>  
// Se crean dos eventos onclick (abrir y cerrar) para aplicar a dos botones input con el fin de  
// abrir y cerrar pestañas del navegador web.  
<input id="boton-abrir" type="button" class="ventana" value="abrir página web Incap"  
onclick="abrir();">  
<input id="boton-cerrar" type="button" class="ventana" value="cerrar ventana"  
onclick="cerrar();">  
</script>
```

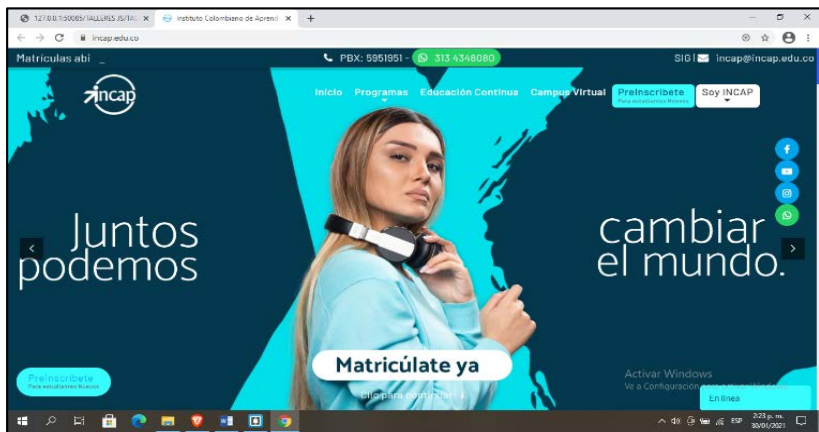
```
var pestana;  
function abrir(){  
    pestana=window.open("https://www.incap.edu.co")  
}  
function cerrar(){  
    pestana.close();  
}  
</script>
```

## RESULTADO FINAL

Vista final, dar clic en abrir pág. Web Incap



## Página Web del Incap



Clic en cerrar ventana, se cierra pág. web Incap





FDEH (Formador dice y estudiante hace)

✓ Completa la instrucción en elemento `<button>` coloca el evento que debería hacer algo cuando alguien hace clic en él.

`<button _____="alert('Estudia Javascript')">Da click acá.</button>`

✓ Completa la instrucción en el elemento `<button>` para cuando se haga clic en el botón, se debe ejecutar la función "cerrar"

`<button _____="_____">Da click acá </button>`

## 2.10 Barra menú lateral Colapsada

Actualmente muchas páginas presentan este tipo de barra lateral de menús colapsada al dar clic en un botón tipo hamburguesa.

**EJEMPLO:**

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="content-type" content="text/html" charset="UTF-8">
<title>EJEMPLO DE SIDEBAR COLLAPSED</title>
<link rel="stylesheet" type="text/css" href="css/mestilo1.css">
<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
integrity="sha384-
50oBUHEmvpQ+1IW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf"
```

```

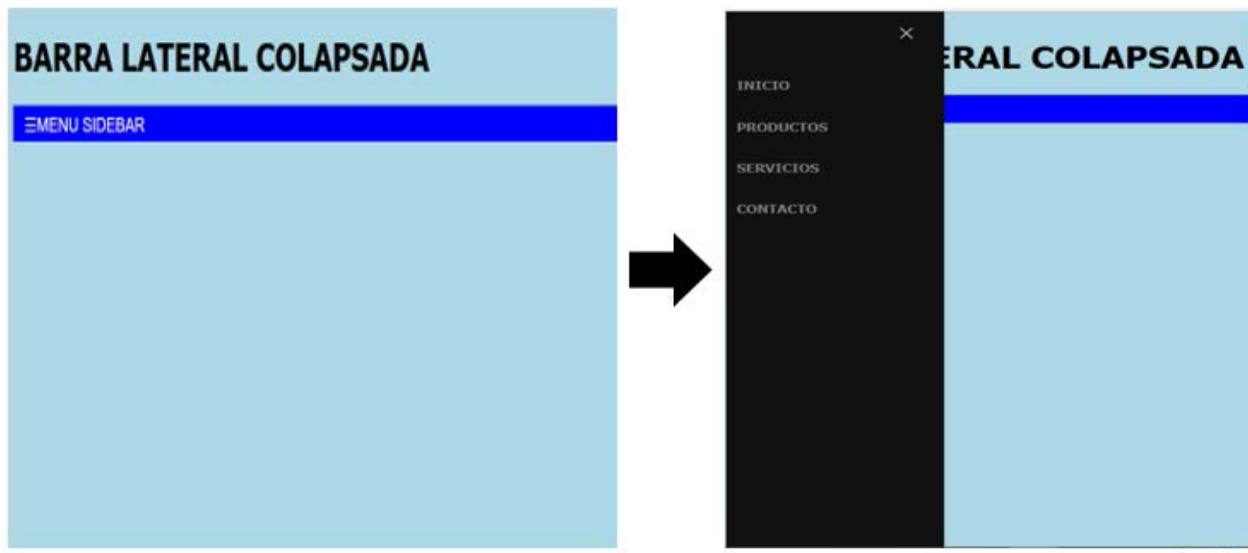
crossorigin="anonymous">
</head>
<body>
<!-- seccion cabecera -->
<header>
<h1> BARRA LATERAL COLAPSADA</h1>
</header>
<!-- seccion menú -->
<nav id="menu">
  <button class="openbtn" onclick="openMenu()">&#9776;MENU SIDEBAR</button>
  <ul class="mainmenu" id="mainMenu">
    <a href="javascript:void(0)" class="closebtn" onclick="closeMenu()">&times;</a>
    <li><a href="#">INICIO</a></li>
    <li><a href="#">PRODUCTOS</a></li>
    <li><a href="#">SERVICIOS</a></li>
    <li><a href="#">CONTACTO</a></li>
  </ul>
</nav>

  <script>
// -- Sección de programacion de funcion para el botón openMenu para abrir la barra lateral
colapsada
  function openMenu() {
    document.getElementById("mainMenu").style.width = "230px";
    document.getElementById("mainMenu").style.height = "800px";
  }
// -- Sección de programacion de funcion para el botón closeMenu para cerrar la barra lateral
colapsada

  function closeMenu() {
    document.getElementById("mainMenu").style.width = "0";
  }
  </script>
</body>
</html>

```

## RESULTADO FINAL



### 2.11 Ventana Modal con funciones JS

En las paginas web actuales se manejan las compras de productos mediante el metodo de carro de compras con este ejemplo vamos a aprender como se crean ventanas modales aplicando funciones de javascript

#### EJEMPLO:

En HTML

```
<!DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type"content="text/html" charset="UTF-8">
  <title></title>
  <link rel="stylesheet" type="text/css" href="css/Productos.css">

</head>
```

```

<body>
  <center>
    <div class="page-content">
      <!-- seccion de contenedor de imágenes donde se coloca la imagen respectiva y el texto
      referente al detalle del producto con su valor correspondiente -->
      <div class="productos">
        <center><h3>Portatil HP |AMD Athlon 4 GB RAM- SSD 256 GB</h3>
        
        <h3>$2,299.000</h3>
      <!-- seccion donde se colocar un botón para dar clic para añadir a carro de compras y
      mostrar la ventana modal correspondiente -->
      <button id="button-add" onclick="carri()" >Añadir carrito</button></center>
      </div>
      <div class="productos">
        <center> <h3>Portatil HP 14"|Intel Core i3- 4 GB RAM- DD 1TB </h3>
        
        <h3>$2,199.000</h3>
        <button id="button-addo" onclick="carro()">Añadir carrito</button></center>
      </div>
    </div>
    <div id="win-modal" class="modalcontainer">
      <div class="modal-content">
        <span class="close">&times;</span><h2>Producto agregado a su carrito de
compras</h2>
        <p>Computador Portatil HP 15,6 Pulgadas 15-gw0003la AMD Athlon Silver- 4 GB RAM-
Disco Estado Sólido 256 GB-Negro <br>Valor compra: $2,299.000</p>
      <!-- Seccion donde se colocar un botón para dar clic para seguir comprando otro producto y
añadirlo al carro de compras correspondiente -->
      <button id="btn">Seguir comprando</button>
    </div>
  </div>

  <div id="win-modalo" class="modalcontainer">
    <div class="modal-content">
      <span class="close">&times;</span><h2>Producto agregado a su carrito de
compras</h2>

```

```

    <p> Computador Portatil HP 14 Pulgadas HP 240G7 Intel Core i3- 4 GB RAM- Disco
    Duro 1TB-Negro<br> Valor compra: $2,199.000</p>
        <button id="btn">Seguir comprando</button>
    </div>
</div>
<script src="js/carritojs.js"></script>
</center>
</body>
</html>

```

## En Javascript

```

function carri() {
    var modal = document.getElementById("win-modal");
    var btn = document.getElementById("button-add");
    var span = document.getElementsByClassName("close")[0];
    var body = document.getElementsByTagName("body")[0];
    // Sección donde se programa cargo de ventas del primer producto y mostrarlo en la
    ventana modal correspondiente -->
    var total=0;
    var carventas=[];
    if (total==0) {
        carventas[0]=2299000;
        var total= carventas[0]+0;
        document.getElementById("carrov").innerHTML=total;
    }
    // Sección donde se programa un botón para añadir producto a carro de compras y mostrar
    la ventana modal correspondiente -->
    btn.onclick = function(){
        modal.style.display = "block";
        body.style.position = "static";
        body.style.height = "100%";
        body.style.overflow = "hidden";
    }
    // Sección donde se programa icono de cerrar la ventana modal correspondiente
    span.onclick = function(){
        modal.style.display = "none";
    }
}

```

```

    body.style.position = "inherit";
    body.style.height = "auto";
    body.style.overflow = "visible";
}
// Sección donde se programa para que al evento de dar clic en la parte externa de la
ventana modal se cierra la misma

window.onclick = function(event) {
    if (event.target == modal) {
        modal.style.display = "none";
        body.style.position = "inherit";
        body.style.height = "auto";
        body.style.overflow = "visible";
    }
}
}
//
function carro() {
    var modal = document.getElementById("win-modalo");
    var btn = document.getElementById("button-addo");
    var span = document.getElementsByClassName("close")[1];
    var body = document.getElementsByTagName("body")[0];
    // Sección donde se programa cargo de ventas del segundo producto y mostrarlo en la
    ventana modal correspondiente -->
    var total2=0;
    var carventas=[];
    carventas[1]=2199000;
    var total2= parseFloat(total)+carventas[1];
    document.getElementById("carrov2").innerHTML=total2;

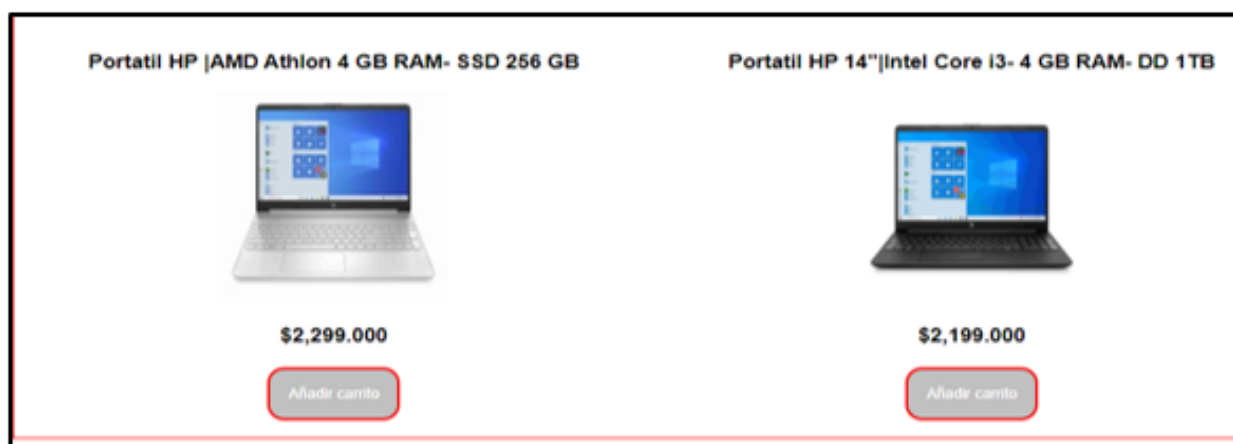
    btn.onclick = function() {
        modal.style.display = "block";
        body.style.position = "static";
        body.style.height = "100%";
        body.style.overflow = "hidden";
    }
}

```

```
span.onclick = function() {  
    modal.style.display = "none";  
  
    body.style.position = "inherit";  
    body.style.height = "auto";  
    body.style.overflow = "visible";  
}
```

```
window.onclick = function(event) {  
    if (event.target == modal) {  
        modal.style.display = "none";  
        body.style.position = "inherit";  
        body.style.height = "auto";  
        body.style.overflow = "visible";  
    }  
}
```

## RESULTADO FINAL



## 2.12 Método AddEventListener()

Es un método que está pendiente de todas las acciones que hace el usuario para que sean escuchadas por el navegador web, estos se efectúan en uno o varios elementos determinados.

### *Sintaxis:*

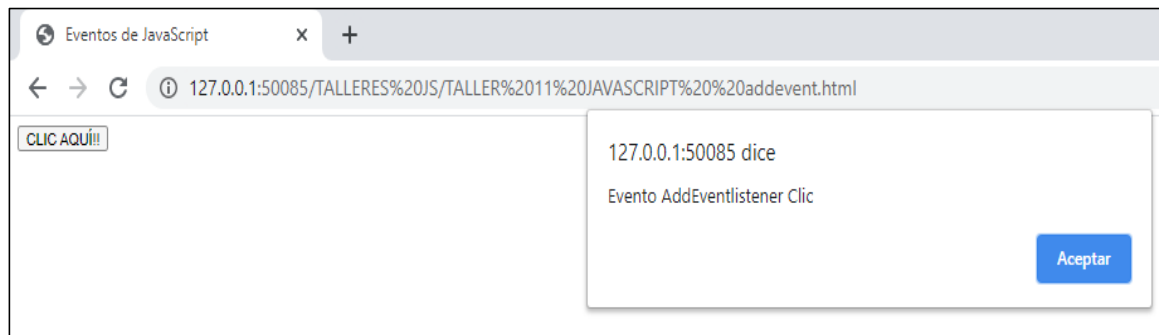
```
Element.addEventListener('evento', función a lanzar(){
})
```



Ejemplo de evento clic para mensaje alerta:

```
// Se crea un boton y se le asigna el método AddEventListener con el evento click al botón y al presionarlo sale un mensaje por ventana
<button id="boton">CLIC AQUÍ!!</button>
<script>
const boton=document.getElementById('boton');
  boton.addEventListener('click',function()=>{
    alert('Evento AddEventListener Clic');
  })
</script>
```

RESULTADO FINAL



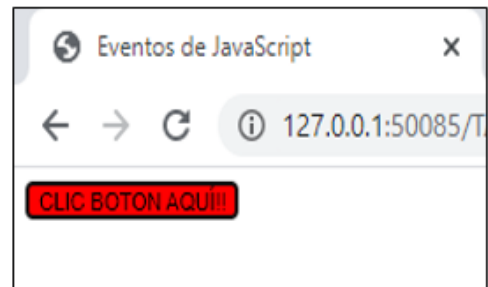
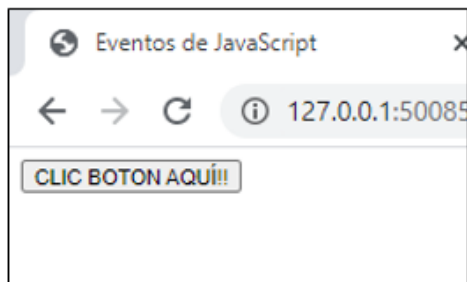
Ejemplo de evento clic para estilo boton:

```
// Se crea un boton y se le asigna a la variable boton todos los estilos referentes al elemento
<button> luego se asigna el método AddEventListener con el evento click al botón y al
presionarlo cambia de color a rojo

<button id="boton-primario">CLIC BOTON AQUÍ!!</button>
<script>
const boton=document.querySelector('button');
boton.addEventListener('click',function(){
  boton.style.backgroundColor="red";
})
```

</script>

## RESULTADO FINAL

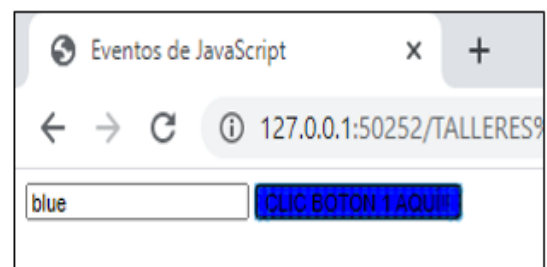
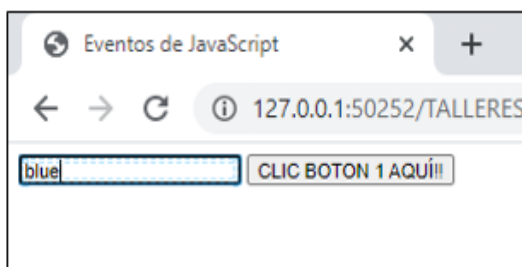


Ejemplo:

// Se crea un input para escribir el color a aplicar al boton y se le asigna a la variable boton todos los estilos referentes a ese elemento <button> luego se asigna el método AddEventListener con el evento click al botón y al presionarlo cambia de color elegido.

```
<input id="color" type="text" placeholder="Ingrese color">
<button class="boton">CLIC BOTON 1 AQUÍ!!</button>
<script>
  const boton=document.querySelector('.boton');
  boton.addEventListener('click',function(){
    var color=document.getElementById("color").value;
    boton.style.backgroundColor=color;
  })
</script>
```

*Atención: el nombre del color debe ser en Inglés.*



## 2.13 Funciones anidadas Método AddEventListener()

En este ejemplo la función carga contiene los métodos listener con los eventos focus y blur para con los inputs las funciones son llamadas clic\_input1 y 2 (focus) y salir\_input 1 y 2 (blur), la idea es aplicar estilos a los < input > del documento HTML.

Para ser ejecutadas las funciones anidadas deben de ser invocadas desde la función inicial.

### EJEMPLO

```
<input type="text" id="dato1">
  <br><br>
  <input type="text" id="dato2">

<script>
  window.onload=carga;
  // En este ejemplo se aplica el evento load para terminar de cargar página web y se invoca la
  función carga, que se utiliza para asignar los métodos addEventListener con los eventos focus
  y blur para los dos inputs. Luego se configura los estilos a aplicarles al dar clic en ellos mismos.

function carga()
{
  document.getElementById('dato1').addEventListener('focus',clic_input1);
  document.getElementById('dato2').addEventListener('focus',clic_input2);
  document.getElementById('dato1').addEventListener('blur',salir_input1);
  document.getElementById('dato2').addEventListener('blur',salir_input2);
}

function clic_input1()
{
  document.getElementById('dato1').style.color='green';
  document.getElementById('dato1').style.backgroundColor='lightblue';
}

function clic_input2()
{
  document.getElementById('dato2').style.color='green';
  document.getElementById('dato2').style.backgroundColor='lightblue';
}
```

```

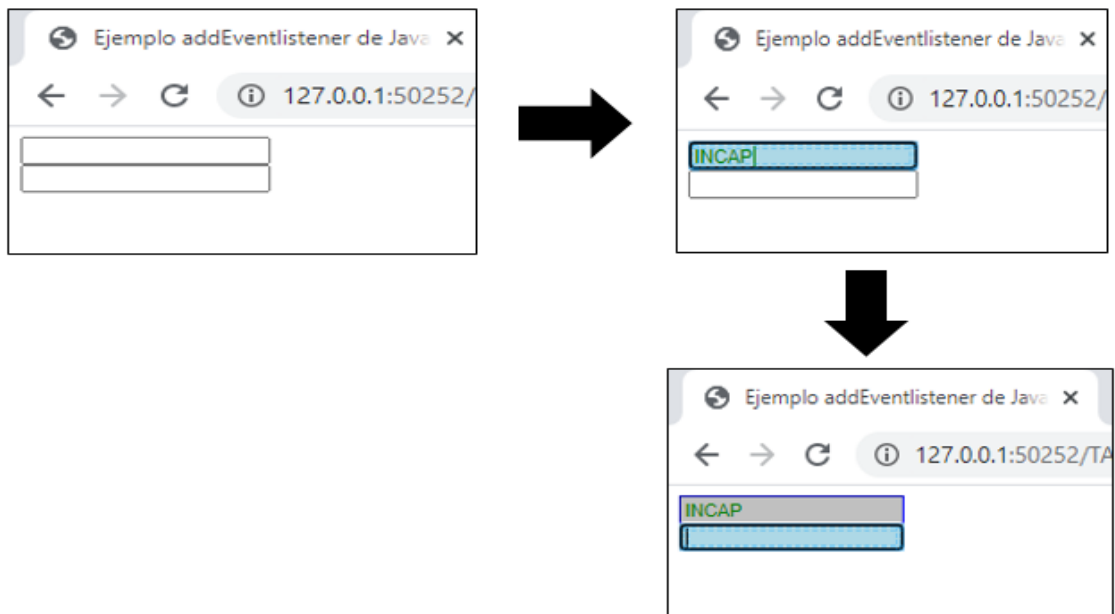
function salir_input1()
{
    document.getElementById('dato1').style.backgroundColor='silver';

}

function salir_input2()
{
    document.getElementById('dato2').style.backgroundColor='silver';

}
</script>

```



## 2.14 Objetos en Javascript

El lenguaje JavaScript no es un POO como tal, aclarando que puede establecer clases con sus propiedades y valores respectivos. Javascript permite crear objetos de determinadas clases, a su vez crear una colección de variables con sus nombres correspondientes.

### 2.14.1 Propiedades y métodos del objeto

Los nombres colocados en la definición de objetos en Js, se llaman **propiedades**. Aclarando que también se pueden definir **métodos de objeto** ya que esta utiliza la creación de una función, dentro del objeto mismo.

Propiedad	Valor
Categoria:	Silla
Marca:	Rimax
Color:	Blanca
Product.venta	<code>function() { return this.categoria + " " + this.marca + " "+this.color</code>

### EJEMPLO:

```
<!DOCTYPE html>
<html>
<body>
  <h1>Ejemplo de manejo de objetos</h1>
  <p id="mensaje"></p>
```

// Se crea un objeto llamado producto con las propiedades categoria, marca y color de producto, se crea una función asociada a la propiedad producto.venta y al final se imprime en pantalla todas las propiedades del objeto // o también se podría haber colocado la siguiente instrucción:

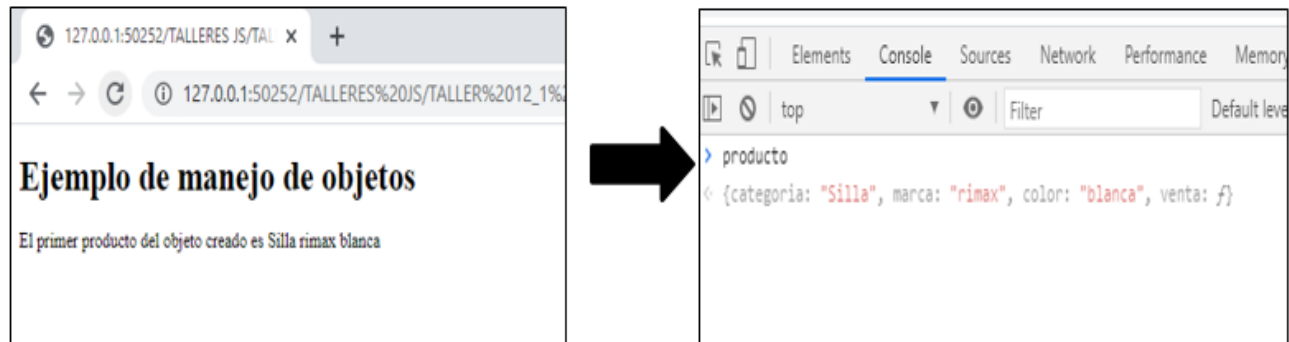
```
document.getElementById("mensaje").innerHTML = "El primer producto del objeto creado es..." + producto.categoria + " " + producto.marca + " " + producto.color;
```

```
<script>
  var producto = new(Object);
  producto.categoria= "Silla";
  producto.marca= "rimax";
  producto.color= "blanca";
  producto.venta = function() {
```

```

    return this.categoria + " " + this.marca + " "+this.color;
};
document.getElementById("mensaje").innerHTML =
"El primer producto del objeto creado es... " + producto.venta();
</script>
</body>
</html>

```



### 2.14.2 Metodo this

Se utiliza el Método de objeto, *this* para hacer referencia al " propietario " de esa propiedad y su valor creado.

La clave para saber que realiza this, es tener claro donde se coloca en JS, para saber qué objeto se le asigna.

En el primer caso this está siendo invocado dentro de un método.

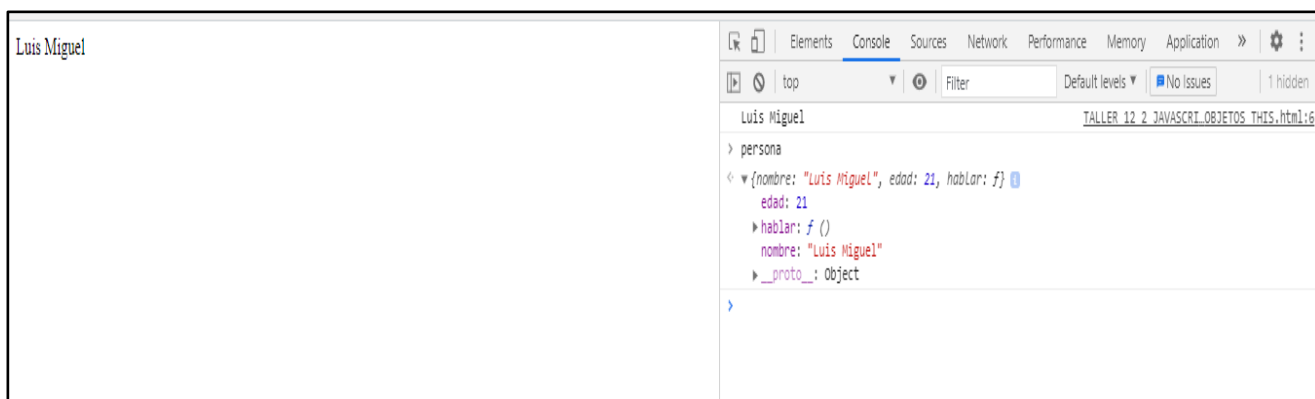
```

let persona = {
  nombre: 'Luis Miguel',
  edad: 21,
  hablar: function () {
    console.log(this.nombre);
  }
};

persona.hablar(); // Luis Miguel

```

### RESULTADO FINAL



## FDEH (Formador dice y el estudiante hace)

Estudiante para afianzar los conocimientos adquiridos debes de realizar los siguientes ejercicios:

- ✓ Coloque para mostrar en alert el valor del nombre de la persona que se encuentra en el objeto creado persona.

```
var persona={nombre: "Carlos", apellido:" Toro",edad:30};  
alert(_____);
```

- ✓ Ahora debes de agregar al objeto persona, la siguiente propiedad y valor: país: Costa Rica.

```
var persona={nombre: "Carlos", apellido:" Toro",edad:"30" _____ };
```

- ✓ Utilice EventListener para asignar un evento onclick al elemento <button>.

```
<button id="dato">Da clic acá</button>  
<script>  
document.getElementById("dato")_____(" _____",abrir);  
</script>
```

## UNIDAD 3

### FORMULARIOS Y VENTANAS DE DIALOGO EN JAVASCRIPT

Diseñar e implementar formularios acordes con las exigencias establecidas por el cliente, utilizando para tal fin eventos, objetos y validación de datos; adicionalmente crea ventanas de dialogo utilizando el lenguaje, así como funciones personalizadas.

#### 2.15 Formularios en JavaScript

Desde JavaScript se puede acceder a los formularios mediante su atributo name o mediante su atributo id; los objetos de los formularios tienen igual manera de acceder a ellos y lógicamente utilizando las funciones DOM, que me permiten acceder a los elementos de un formulario de forma directa.

Cada elemento del formulario posee las siguientes propiedades:

- ✓ **type**: indica el tipo como por ejemplo input, text, button, checkbox, textarea, select-one o select-multiple etc.
- ✓ **form**: con esta propiedad se hace referencia directamente al formulario y por ende a los elementos de este.
- ✓ **name**: obtienen el nombre del objeto y como será identificado en el sistema.
- ✓ **value**: permite leer y modificar el contenido textual o numérico si es el caso del objeto en cuestión; por ejemplo, en los objetos de tipo texto como text o textarea; para los **lo** objetos como los botones se estaría haciendo mención del contenido textual del botón.

Los eventos que más utilizaremos respecto a los formularios son:

- ✓ **onclick**: cuando damos click en un objeto del formulario.
- ✓ **onchange**: se produce al momento de cambiar el valor del objeto y esto ocurre cuando el foco abandona el objeto.
- ✓ **onfocus**: este evento se produce cuando un objeto toma el foco o es seleccionado.

#### 2.16 Manejo de Controles visuales en un formulario

**Obtener el valor de los campos de un formulario:** la mayoría de los procesos que se realizan con los formularios son los de capturar y modificar el contenido textual de un objeto. A continuación, veremos cómo se realiza el proceso con los objetos más utilizados.

##### 2.16.1 Cuadro de texto y textarea:



Los valores alojados en estos elementos se obtienen y modifican mediante la propiedad value.

Miremos un ejemplo básico para los dos objetos:

Así se define el objeto text y textarea en la página web dentro de un formulario:

```
<input type="text" id="texto1"/>  
<textarea id="articulo"></textarea>
```

Así se captura la información contenida en el cuadro de texto y en el textarea, y se guardan en la variable valor1 y valor2 respectivamente:

```
//Obtener valor de cuadro de texto//
```

```
var valor1=document.getElementById("texto1").value;
```

```
//Obtener valor de un textarea//
```

```
var valor2= document.getElementById("articulo").value;
```

```

<body>
<h1>Obtener el valor de un radio button</h1>
<form id="form1">
  Nombre:<br><input type="text" name="nombre" value="" id="nombre" class="formulario">

  <!--<p><input type="checkbox" name="acepto" id="acepto" class="formulario_check"> Acepto el contrato</p-->

  <p>Deacuerdo: Si<input type="radio" name="deacuerdo" value="si"> No<input type="radio" name="deacuerdo" value="no"></p>

  <p>
    <select name="seleccion" id="seleccion" class="formulario_select">
      <option value="1">primera</option>
      <option value="2">segunda</option>
    </select>
  </p>
</form>
<input type="button" value="Que valor hay en el radio button" onclick="capturar()">
<div id="resultado"></div>

<p><a href="http://www.lawebdelprogramador.com">http://www.lawebdelprogramador.com</a></p>
</body>
</html>

```

Activar  
Ve a Con

El código anterior crea el formulario que permite capturar los datos.

### 2.16.2 Botones de opción (radiobutton):

En este objeto solo se selecciona una única opción y en muchos casos queremos saber cuál de ellos fue seleccionado; para ello recurrimos a la propiedad `checked` que devuelve verdadero si el botón de opción fue seleccionado y falso en caso contrario.

Miremos un ejemplo completo respecto a la programación de *los botones de opciones y del cuadro de texto*:

Para hacer una lectura por todos los botones de opciones creados se debe de asignar a una variable tipo array el método `document.getElementsByName` (todos los botones de opción indicados deben de tener la misma clase como nombre (`class= "name"`) y luego se declara un ciclo repetitivo teniendo como limite la longitud total del array método (`length`), luego se coloca un condicional preguntando si el boton del índice respectivo ha sido seleccionado.

```
function capturar()
{
    var resultado="ninguno";

    var nombrel=document.getElementById('nombre').value;
    var Aceptar=document.getElementsByName("deacuerdo");
    // Recorremos todos los valores del radio button para encontrar el
    // seleccionado
    for(var i=0;i<Aceptar.length;i++)
    {
        if(Aceptar[i].checked)
            resultado=Aceptar[i].value;
    }

    document.getElementById("resultado").innerHTML=nombrel+" "+" \
    Acepto contrato?: "+resultado;
}
</script>
```

El código anterior corresponde a la función que realiza todo el proceso. Captura el nombre de la persona en un formulario, y de termina al final si dicha persona acepto o no el contrato.

```

<style>
    form    {width:250px;
              height:180px;
              border:1px solid #ccc;
              padding:10px;
              }
</style>

```

El código anterior da un formato sencillo al formulario, utilizando reglas CSS.

### 2.16.3 Casillas de verificación (checkbox):

Las casillas de verificación tienen una forma similar de capturar su información a la de los radiobutton, con la diferencia que se pueden seleccionar varias casillas a la vez y cada checkbox se debe comprobar de manera independiente y no en conjunto como los radiobutton. Esta característica ya la conocen los estudiantes pues tanto los botones de opción como las casillas de verificación tienen diferentes objetivos que se han explicado en sesiones anteriores.

Programación de *checkbox*:

```

var aceptol=document.getElementById('acepto');

if(aceptol.checked){
    alert('mensaje de aceptacion');
}
else{
    alert('no aceptado');
}

```

El código anterior nos muestra de manera sencilla como podemos verificar si se ha seleccionado una casilla de verificación o no.

### 2.16.4 Cuadros combinados (select):

```

var lista = document.getElementById("opciones");
// Obtener el valor de la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].value;
// Obtener el texto que muestra la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].text;
alert(valorSeleccionado);

```

Si observamos el ejemplo, manejamos en él varias propiedades como:

**Options**, que corresponde a un array creado automáticamente por el navegador para las listas desplegables y adicionalmente contiene la referencia a todas las opciones de esa lista.

*Por ejemplo, si queremos la segunda opción de una lista tendríamos:*

```
Var lista= document.getElementById("opciones").options[1];
```

**selectedIndex**, cuando seleccionamos una opción el navegador toma el valor de esta propiedad que corresponde al índice de la opción seleccionada. Recuerda que: la primera posición del array es 0 , luego 1 , luego 2, y así sucesivamente.

## 2.17 Validación de un formulario en JS

A continuación, se darán los lineamientos básicos para validar un formulario respecto a los controles más utilizados en el mismo.

### 2.17.1 Validación de campos obligatorios:

en muchos formularios, podríamos decir que en la mayoría existen campos que son obligatorios o esenciales; por tal razón debemos asegurarnos de tres aspectos fundamentales: que no contengan valores nulos, que el campo no se encuentre vacío y no estén llenos de espacios en blanco. Teniendo en cuenta lo anterior veamos el código pertinente:

```

var txtNombre= document.getElementById("texto").value;
if(txtNombre == null || txtNombre.length== 0 ||
/^\\s+$/.test(txtNombre)){
alert('ERROR: El campo nombre es obligatorio');
return false; }

```

### 2.17.2 Validación de campos numéricos:

En un formulario los campos numéricos son algo más exigentes que los anteriores ya que solo

podemos permitir que solo sean ingresados únicamente números; una función que ya se mencionó en apartados anteriores y que será de gran utilidad es `isNaN`. Veamos el código sugerido para la validación de campos numéricos.

```
var txtEdad= document.getElementById("edad").value;
if(txtEdad == null || txtEdad.length == 0 || isNaN(txtEdad)){
alert('ERROR: Debe ingresar una edad');
return false;
}
```

### 2.17.3 Validación de campos de correo electrónico:

En muchos formularios se solicitan correos electrónicos y obviamente necesitamos validar la correcta digitación de esta información; para ello se utilizará una expresión regular que nos permitirá determinar su correcta escritura. Veamos el código sugerido:

```
var txtCorreo= document.getElementById("email").value;
if(!(/^\S+@\S+\.\S+$/).test(txtCorreo)){
alert('ERROR: Debe escribir un correo válido');
return false;
}
```

### 2.17.4 Validación de cuadros combinados:

para validar listas de opciones como un cuadro combinado, debemos tener en cuenta que cada elemento de la lista tiene un índice. Veamos entonces el código sugerido:

```
var cmbSelector= document.getElementById("selectbox").value;
if(cmbSelector == null || cmbSelector == 0){
alert('ERROR: Debe seleccionar una opción del combo box');
return false;
}
```

### 2.17.5 Validación de casillas de verificación (checkbox):

Las casillas de verificación pueden estar o no seleccionadas y ellas devuelven un valor lógico que puede ser falso (no seleccionado) o verdadero(seleccionado). *Recuerda que: no se necesita su valor sino si está habilitada la casilla* , Miremos cual sería el código sugerido:

```
var chkEstado= document.getElementById("ofertas");
if(!chkEstado.checked){
alert('ERROR: Debe seleccionar el checkbox');
return false;
}
```

```
}
```

### 2.17.6 Validación de campos tipo fecha:

Para estos campos lo que debemos hacer es negar la función isNaN ya que ella devuelve falso si se encuentran números en el campo determinado. Veamos el código sugerido:

```
var txtFecha= document.getElementById("fecha").value;
  if(!isNaN(txtFecha)){
alert('ERROR: Debe elegir una fecha');
return false;
}
```

### 2.17.7 Validación de botones de opción (radio button):

Los radios button a diferencia que los checkbox permiten la selección de uno solo de un conjunto de estos elementos, por ende, la programación para su validación requiere de algo adicional como es utilizar un ciclo que me permita recorrer el arreglo de opciones que el sistema crea automáticamente, *Recuerda que: Se necesita un array para saber la opción que está seleccionada, y si está habilitada si/no.* Veamos el código sugerido:

```
let rbtEstado= document.getElementsByName("opciones");

for(var i = 0; i < rbtEstado.length; i++){
  if(rbtEstado[ i ].checked){
banderaRBTN = true;
break;
}
}
if(!banderaRBTN){
alert('ERROR: Debe elegir una opción de radio button'); return false;
}
```

### 2.17.8 VALIDAR FORMULARIO BASICO

Se puede validar un formulario de una forma básica sin utilizar lenguaje Javascript para ello utilizamos el atributo *pattern* y los valores de validación respectivos a cada control visual veamos cómo se puede hacer:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <!-- <link rel="stylesheet" href="css/f2estilos.css">-->
  <title>EJERCICIO 1 DE VALIDACION DE FORMULARIO BASICO</title>
</head>
<body>
  <div class="contenedor">
    <div class="cabecera">
      <h2>Formulario de registro de cliente</h2>
    </div>
    <!--En esta línea se configura el formulario para que tenga un acción de enviar por correo de los datos como tipo texto plano-->
    <form class="form_register" id="formulario"
action="mailto:heladio.polo@incap.edu.co" enctype="text/plain" method="post"
>
      <!--En esta línea se configura el ingreso del usuario del cliente validado con los parámetros letras mayúscula o minúscula, guion al piso o punto con longitud desde seis a 10 caracteres-->

      <input type="text" id="usuario" placeholder="Ingrese usuario"
required autofocus autocomplete="off" pattern="[A-Za-z0-9_.]{6,10}">
      <br>
      <!--En esta línea se configura el ingreso del nombre completo del cliente validado con los parámetros letras mayúscula o minúscula y un espacio en blanco, con longitud desde dos a 30 caracteres-->
```



```
<input type="text" id="nombre" placeholder="Ingrese nombre y  
apellido" required autofocus autocomplete="off" pattern="[A-Za-z ]{2,30}">
```

```
<br>
```

```
<input type="email" id="email" placeholder="Ingrese Email"  
required autocomplete="off" >
```

```
<br>
```

*<!--En esta línea se configura el ingreso del número telefónico del cliente validado con los  
parámetros numéricos del 0 al 9, con longitud desde seis a 10 números-->*

```
<input type="text" id="telefono" placeholder="Ingrese Telefono"  
required autocomplete="off" pattern="[0-9]{6,10}">
```

```
<br>
```

*<!--En esta línea se configura el ingreso de la edad del cliente validado con los parámetros  
numéricos desde 18 hasta 100-->*

```
<input type="number" id="edad" placeholder="Ingrese Edad"  
required autocomplete="off" min="18" max="100">
```

```
<br>
```

```
<input type="checkbox" id="terminos" required >Acepto términos y  
condiciones
```

```
<br>
```

```
<div class="btn_accion">
```

*<!--En esta línea se configura el botón de envío con el evento submit ligado a la acción  
mailto:" " y el botón de cancelar con el evento reset para limpiar todos los input con datos  
y permitir un nuevo envío desde 0-->*

```
<button class="btn_enviar" type="submit" >ENVIAR</button>
```

```
<button class="btn_limpiar" type="reset" >CANCELAR</button>
```

```
</div>
```

```
</form>
```

```
</div>
```

```
</body>
```

```
</html>
```

✓ Debes de crearle los estilos al formulario y debes de grabar el documento HTML del formulario en la carpeta PROYECTO CLASE y la subcarpeta html con el nombre **formulario\_validabasico.html** y el documento de estilos css en la subcarpeta de css con el nombre **FREGISTER.css**

RESULTADO FINAL



Formulario de registro de cliente

✉ Ingrese usuario

👤 Ingrese nombre y apellido

✉ Ingrese Email

☎ Ingrese Telefono

👤 Ingrese Edad

☐ Acepto términos y condiciones

ENVIAR CANCELAR

## 2.18 EXPRESIONES REGULARES EN JS

Una expresión regular es una herramienta muy importante que se utiliza para definir un patrón de búsqueda y poder validar ingreso de datos de forma correcta, JavaScript maneja las expresiones regulares con el objeto RegExp.

### 2.18.1 METODOS DE EXPRESIONES REGULARES

Método	Función
test(string)	Se utiliza para comprobar si el String concuerda con el patrón de expresión regular su resultado es un booleano (verdadero o falso)
Exec(string)	Se utiliza para aplicar una búsqueda del patrón definido en el string su resultado es un array.

### 2.18.2 Patrones de búsquedas de expresiones regulares

Se utiliza para definir cuáles son los parámetros de comparación y validación de los datos ingresados en un control visual.

Delimitadores:

***Delimitador    Función***

/..... /	Define el patrón de búsqueda establecido en JS
^	Define el comienzo de un patrón si está al comienzo de un corchete [ significa que comience por esos números o letras requeridas.
\$	Define el final de un patrón de búsqueda va antes del delimitador /

Conjunto de caracteres, se utilizan corchetes:

***Expresión    Función***

[abc]	Concordar con letras que están en los corchetes.
-------	--

<i>[0-9]</i>	Concordar con números del 0 al 9 que están en los corchetes.
<i>(x/y)</i>	Concordar el patrón de x o el patrón de y que están en los paréntesis.
<i>[^abc]</i>	No exista cualquiera de los caracteres del interior de los corchetes.
<i>[^0-9]</i>	No existan números del 0 al 9.

Caracteres especiales:

***Carácter especial    Función***

<i>ld</i>	Concordar con un dígito
<i>ls</i>	Concordar con un espacio en blanco
<i>lw</i>	Concordar con una letra o dígito

Cuantificadores:

***cuantificador    Función***

<i>string+</i>	Concordancia puede aparecer mínimo 1 o más vez
<i>string*</i>	Concordancia puede aparecer 0 o más veces.

<i>string?</i>	Concordancia puede aparecer o no puede aparecer.
<i>{n}</i>	Longitud de n caracteres.
<i>{n,}</i>	Longitud de n o más caracteres (sin límite)
<i>{n1,n2}</i>	Longitud de caracteres de n1 hasta n2

### 2.18.3 Validación avanzada de formularios

Es importante crear en un formulario las validaciones de todos los campos creados en él con el fin de verificar que al momento de enviar los datos a un servidor web éstos se envíen de forma clara, precisa y con integridad es decir sin campos nulos, incompletos o no acordes a las reglas de ingreso de datos.

#### EJEMPLO:

##### En HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="css/Formavanz_estilos.css">
  <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css"
integrity="sha384-
50oBUHEmvpQ+1IW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf"
crossorigin="anonymous">
  <title>Formulario valida avanzado</title>
</head>
<body>
<div class="container">
  <div class="header">
```

```

    <h3>VALIDAR INGRESO CUENTA</h3>
</div>
<form id="form" class="form" action="">
  <div class="form-control">
    <label for="usuario">USUARIO</label>
    <input type="text" placeholder="ingrese usuario" id="usuario">
    <i class="fas fa-check-circle"></i>
    <i class="fas fa-exclamation-circle"></i>
    <small>mensaje de error</small>
  </div>
  <div class="form-control">
    <label for="email">EMAIL</label>
    <input type="text" placeholder="ingrese cuenta@servidor.com" id="email">
    <i class="fas fa-check-circle"></i>
    <i class="fas fa-exclamation-circle"></i>
    <small>mensaje de error</small>
  </div>
  <div class="form-control">
    <label for="password">CONTRASEÑA</label>
    <input type="password" placeholder="ingrese contraseña" id="password">
    <i class="fas fa-check-circle"></i>
    <i class="fas fa-exclamation-circle"></i>
    <small>mensaje de error</small>
  </div>
  <div class="form-control">
    <label for="password2">REPETIR CONTRASEÑA</label>
    <input type="password" placeholder="Vuelva a ingresar contraseña"
id="password2">
    <i class="fas fa-check-circle"></i>
    <i class="fas fa-exclamation-circle"></i>
    <small>mensaje de error</small>
  </div>
  <button>ENVIAR</button>
</form>
</div>
<script src="js/validarformavanz.js"></script>
</body>

```

```
</html>
```

### En Javascript:

```
const form= document.getElementById("form");
const usuario= document.getElementById("usuario");
const password= document.getElementById("password");
const password2= document.getElementById("password2");

form.addEventListener('submit', e =>{
  e.preventDefault();
  validainputs();
});
function validainputs() {
  const usuarioval= usuario.value.trim();
  const emailval= email.value.trim();
  const passwordval= password.value.trim();
  const password2val= password2.value.trim();
  if (usuarioval === "") {
    ocErrorform(usuario, "no puede usuario estar vacío");
  }else{
    exitosform(usuario);
  }
  if (emailval === ""){
    ocErrorform(email,"no se puede dejar email en blanco");
  }else if (!valEmail(emailval)){
    ocErrorform(email,"no ingreso, email no valido");
  }else{
    exitosform(email);
  }
  if (passwordval === ""){
    ocErrorform(password,"no se puede dejar password en blanco");
  }else if (!valPassword(passwordval)){
    ocErrorform(password,"no ingreso, password no valido");
  }else{
    exitosform(password);
  }
}
```

```

}
if (password2val === ""){
  ocErrorform(password2,"no se puede dejar password en blanco");
}else if ( passwordval!==password2val){
  ocErrorform(password2,"error las contraseñas no coinciden..corregir");
}else{
  exitosform(password2);
}
}

function ocErrorform (input, message){
  const formControl=input.parentElement;
  const small=formControl.querySelector("small");
  formControl.className="form-control error";
  small.innerText =message;
}

function exitosform (input){
  const formControl=input.parentElement;
  formControl.className="form-control success";
}

function valEmail(email){
return /^[a-z0-9_\.-]+@[a-z\.-]+\.[a-z\.-]{2,6}$/i.test(email);
}

function valPassword(password){
  return /^[a-z0-9_-]{6,10}$/i.test(password);
}

```

## RESULTADO FINAL



The diagram illustrates the state change of a login form titled "VALIDAR INGRESO CUENTA".

**Initial State (Left):** The form contains four input fields, each with a red border and a red error icon:

- USUARIO:** "ingrese usuario" (Error: "no puede usuario estar vacío")
- EMAIL:** "ingrese cuenta@servidor.com" (Error: "no se puede dejar email en blanco")
- CONTRASEÑA:** "ingrese contraseña" (Error: "no se puede dejar password en blanco")
- REPETIR CONTRASEÑA:** "Vuelva a ingresar contraseña" (Error: "no se puede dejar password en blanco")

**Final State (Right):** The form is filled with valid data, indicated by green borders and green checkmarks:

- USUARIO:** "heladio.polo" (Valid)
- EMAIL:** "poloheladio@gmail.com" (Valid)
- CONTRASEÑA:** "\*\*\*\*\*" (Valid)
- REPETIR CONTRASEÑA:** "\*\*\*\*\*" (Valid)

A large black arrow points from the initial state to the final state.

✓ Debes de crearle los estilos al formulario y debes de grabar el documento HTML del formulario en la carpeta PROYECTO CLASE y la subcarpeta html con el nombre **formulario\_validavanza.html** y el documento de estilos css en la subcarpeta de css con el nombre **Formavanz\_estilos.css** y el documento de javascript en la subcarpeta de js con el nombre **validarformavanz.js**

*Recuerda: enlazar el archivo de estilos css con el documento HTML **INDEX.html** y además enlazarlo con formulario avanzado anterior en la línea:*

```
<p><a href="html/formulario_validavanza.html">Registrese aquí</a></p>
```

The footer section has a blue border and contains the following text:

[Registrese aquí](#)

Derechos Reservador © nombre completo estudiante

Ahora vamos a crear una validación de datos al formulario de registro cliente de nuestra página web vista en el trimestre pasado en HTML y CSS, con el fin de depurar datos antes de enviarlos al controlador.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" type="text/css" href="css/festilos.css">
  <title> FORMULARIO</title>
</head>
<body>
  <header><center>
    <h1>Formulario de registro cliente </h1>
  </center>
</header>
<br>
<section>
  <form>
    <fieldset>
      <label for="nombre">Nombre completo:
      </label>
      <input id="nombre" type="text" placeholder="Ingrese su nombre y apellido"
required autofocus autocomplete="off">
      <br><br>
      <label for="edad">Edad:
      </label>
      <input id="Edad" type="text" placeholder="Ingrese su edad" required
autocomplete="off">
      <br><br>
      <label for="Direccion">Dirección:
      </label>
      <input id="Dir" type="text" placeholder="Ingrese su Dirección" required
autocomplete="off">
      <br><br>
      <label for="Email">Email:
```

```

        </label>
        <input id="Email" type="email" placeholder="Ingrese su email" required
autocomplete="off">
        <br><br>
        <label for="ofertas">
        No deseo recibir ofertas via email: </label>
        <br>
        <input name="ofertas" id="ofer1" type="checkbox">Si<br>
        <input name="ofertas" id="ofer2" type="checkbox">No<br>
        <br>
        <br>
        <label for="SMS">No deseo recibir ofertas via SMS: </label>
        <br>
        <input name="si" type="checkbox">Si<br>
        <input name="no" type="checkbox">No<br>
        <br>
        <label for="fecha">Fecha</label>
        <input type="date" name="fecha" id="fecha"><br><br>
        <label for="hora">Hora</label>
        <input type="time" name="horas" id="horas"><br><br>
        <label for="coment">Tu comentario:</label>
        <br>
        <textarea name="mensaje" id="mensaje" rows="5" cols="40"></textarea>
    </fieldset>
<!-- Se crea los eventos onclick para los botones ENVIAR para validar los datos ingresados y
después enviarlos y al boton CANCELAR para borrar datos ingresados en formulario al dar clic
- - >
    <button type="button" onclick="enviarform()">ENVIAR</button>
    <button type="button" onclick="limpform()" >CANCELAR</button>
    </form><br><br>
    <div id="error"></div>
<br>
<br>
</section>
<footer>
<p><a href="html/ formulario_validavanza.html">Registrese aquí</a></p>
    Derechos Reservados &copy; nombre completo estudiante

```

```
</footer>
```

```
<script type="text/javascript">
```

```
// Se crea función limpform para borrar datos ingresados en formulario al dar clic en boton  
Cancelar
```

```
function limpform(){  
    window.location.reload();  
}
```

```
// Se crea función enviarform primero se declaran variables del formulario de los campos o  
controles visuales para permitir su validación y posterior envío de datos.
```

```
function enviarform(){  
    var nombre,edad, dir, contra, contra2,error,expcorreos;  
    var nombre = document.getElementById("nombre");  
    var edad = document.getElementById("Edad");  
    var dir = document.getElementById("Dir");  
    var email = document.getElementById("Email");  
    var contra = document.getElementById("contra");  
    var contra2 = document.getElementById("contra2");  
    var error = document.getElementById("error");  
    var check1= document.getElementById("ofer1");  
    var check2= document.getElementById("ofer2");  
    var fecha = document.getElementById("fecha");  
    var horas = document.getElementById("horas");  
    var mensaje= document.getElementById("mensaje");  
    var errorf= 0;
```

```
// Se crea variable con propiedad style.color del mensaje de error con estilo de  
color rojo y se define variable tipo array para agregar todos los mensajes de error que se  
presenten al validar datos.
```

```
    error.style.color="red";  
    var mensajeserror=[];
```

```
// Se crean las validaciones de cada uno de los controles visuales o campos para dar  
integridad a los datos si hay fallas genera un mensaje de error
```

```
    if (nombre.value==""){  
        mensajeserror.push('falta ingresar tu nombre completo !!');  
        errorf=1;
```

```

    }
    if (nombre.value!="") {
        nombre = document.getElementById("nombre").value;
    }

    if (isNaN(edad.value) || edad.value == ""){
        mensajeserror.push('la edad no es un número o falta edad !!');
        errorf=1;
    }
    if (!isNaN(edad.value)){
        edad = document.getElementById("Edad").value;
    }
    if (dir.value=== ""){
        mensajeserror.push('falta dirección !!');
        errorf=1;
    }
    if (dir.value!=""){
        dir = document.getElementById("Dir").value;
    }

```

// Se crea la variable de expresión regular para validar el ingreso del email del cliente y dar integridad a los datos si hay fallas genera un mensaje de error

```

//var expcorreo = /^[a-zA-Z0-9_\.]+\@[da-zA-Z0-9]+\.[da-zA-Z0-9-]{2,6}\.[da-zA-Z]{2}$/;

```

```

    var expcorreo = /^[a-z0-9_\.]+\@[a-z\.-]+\.[a-z\.]{2,6}$/;
    if (!expcorreo.test(email.value)) {
        mensajeserror.push('correo no valido !!');
        errorf=1;
    }
    if (expcorreo.test(email.value)) {
        email = document.getElementById("Email").value;
    }
    if ( check1.checked == false && check2.checked == false){
        mensajeserror.push('completa campo ofertas !!');
        errorf=1;
    }
    if ( check1.checked == true){

```

```

    check1= document.getElementById("ofer1").value;
    }

    if ( check2.checked == true){
        check2= document.getElementById("ofer2").value;
    }

    if (fecha.value == ""){
        mensajeserror.push('falta fecha por ingresar !!');
        errorf=1;
    }

    // Se establece un mensaje por consola de enviando datos de formulario

    console.log('validando y enviando datos formulario');

    // Se asigna al <div> error un contenido con la propiedad innerHTML con el fin de
    //mostrar todos los mensajes de error se les agrega una coma ( ,) más un espacio en blanco

    error.innerHTML=mensajeserror.join(', ');
    return false
}
</script>
</body>
</html>

```

## RESULTADO FINAL

## Formulario de registro cliente

Nombre completo:

Edad:

Dirección:

Email:

No deseo recibir ofertas via email:

☐ Si

☐ No

No deseo recibir ofertas via SMS:

☐ Si

☐ No

Fecha

Hora

Tu comentario:

ENVIAR

CANCELAR

*Falta ingresar tu nombre completo !!, la edad no es un número o falta edad !!, falta dirección !!, correo no valido !!, completa campo ofertas !!, falta fecha por ingresar !!, falta hora por ingresar !!, falta mensaje por ingresar !!*

[Regístrate aquí](#)

Derechos Reservados © nombre completo estudiante

## FDEH (formador dice y estudiante hace)

Estudiante junto con tu formador para reafirmar los conocimientos adquiridos debes de realizar lo siguiente:

- ✓ Continuar con la validación de campos checkbox de no deseo recibir ofertas via SMS
- ✓ Continuar con la validación de campos de fecha y hora
- ✓ Continuar con la validación de campo de textarea de comentario
- ✓ Recordar que para que quede nuestro formulario bien debe de ser validados todos los campos y al final debe de mostrar una ventana de dialogo escribiendo el mensaje " Enviando datos del formulario".

## EDH (Estudiante dice y hace)

✓ Estudiante ahora vamos a cargar el formulario de registro con validaciones básicas ya hecho anteriormente para seguir afianzando este importante tema debes de aplicar lo siguiente:

✓ Aplicar validación de datos a todos los campos que componen el formulario y a su vez se requiere aplicar las siguientes expresiones regulares:

- ✓ Para campo **usuario**: /^[a-zA-Z0-9\_-]{6,12}\$/
- ✓ Para campo **email**: /^[a-z0-9\_\.-]+@[a-z\.-]+\.[a-z\.-]{2,6}\$/
- ✓ Para campo **telefono**: /^\d{7,14}\$/
- ✓ Para campo **password**: /^[a-z0-9\_-]{6,10}\$/ (campo opcional)

✓ Debes de crearle los estilos al formulario y debes de grabar el documento HTML del formulario en la carpeta PROYECTO CLASE y la subcarpeta html con el nombre **formulario\_validabasico.html** y el documento de estilos css en la subcarpeta de css con el nombre **FREGISTER.css**

## 3 FRAMEWORK (LIBRERÍAS) EN JS

### 3.1 Qué son las librerías

En desarrollo web, una librería es un recurso o conjunto de recursos que se utilizan para facilitar el desarrollo de documentos Html. Las librerías, también son denominadas "frameworks", ellas consisten en instrucciones de código que por lo general se colocan en la etiqueta <head> al



principio de un documento web.

Las librerías más utilizadas con javascript, son jQuery y MooTools. Cada una de ellas tiene sus características y su manera de trabajar en la página.

### jQuery

Es una librería cuyo objetivo es hacer de tu trabajo en programación web más sencilla con javascript. Para lograr este objetivo se fundamenta en el objeto jQuery el cual permite acceder a los diferentes elementos HTML y lograr aplicarles diferentes tipos de métodos y propiedades que conllevan a ahorrar tiempo y trabajo en la programación con javascript.

### Mootools

Es una librería javascript, que es más completa y a la vez más complicada de utilizar que JQuery.

A continuación, vamos a aprender cómo se crea un visor de slider de imágenes.

## 3.1 VISOR DE IMÁGENES CON JS

### EJEMPLO

Estudiante vamos a crear un visor de imágenes de tipo slider automático, utilizando JQuery y bootstrap.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Ejemplo Bootstrap JS- Slider carrusel</title>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
  <script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js"></script>
<!-- estilos css carrusel Acá se crean los estilos del formulario y su contenido como por
ejemplo tamaño del contenedor de imágenes, márgenes, color de letra de títulos y parrafo.--
```

```

>
<style>
.carousel-inner > .item > img,
.carousel-inner > .item > a > img {
  width: 55%;
  margin: auto;
}
p,h3{
  color:black;
}
</style>
</head>
<body>

<div class="container">
  <br>
  <div id="slide_Carousel" class="carousel slide" data-ride="carousel">
    <!-- Indicators Aquí se crean el carrusel tipo slider con una lista luego las clases con el fin de
hacer enlazar los botones a las imágenes slides del formulario-->
    <ol class="carousel-indicators">
      <li class="slide1 active"></li>
      <li class="slide2"></li>
      <li class="slide3"></li>
      <li class="slide4"></li>

    </ol>

    <!-- imagenes slides Aquí se colocan las imágenes que tendrá el carrusel tipo slide con un
titulo y un texto referente. -->
    <div class="carousel-inner" role="listbox">

      <div class="item active">
        
        <div class="carousel-caption">
          <h3>Diseño web 1</h3>
          <p>Diseño web es espectacular.</p>
        </div>

```

```

</div>
<!-- imagenes slides Aquí se colocan las imágenes que tendrá el carrusel tipo slide con un
titulo y un texto referente. -->
<div class="item">
  
  <div class="carousel-caption">
    <h3>Diseño web 2</h3>
    <p>HTML estudia ya es importante.</p>
  </div>
</div>
<!-- imagenes slides Aquí se colocan las imágenes que tendrá el carrusel tipo slide con
un titulo y un texto referente. -->
<div class="item">
  
  <div class="carousel-caption">
    <h3>Diseño web 3</h3>
    <p>Javascript estudia ya es excelente.</p>
  </div>
</div>
<div class="item">
  
  <div class="carousel-caption">
    <h3>Diseño web 4</h3>
    <p>CSS3 estudia ya es excelente.</p>
  </div>
</div>

</div>

<!--controles de movimiento derecho e izquierda Aquí en esta seccion se crean los botones
de movimiento izquierda(prev) y derecha (next) ubicados a ambos lados de las imágenes-->
<a class="left carousel-control" href="#slide_Carousel" role="button" data-slide="prev">
  <span class="glyphicon glyphicon-chevron-left" aria-hidden="true"></span>
  <span class="sr-only">Anterior</span>
</a>
<a class="right carousel-control" href="#slide_Carousel" role="button" data-
slide="next">

```

```

    <span class="glyphicon glyphicon-chevron-right" aria-hidden="true"></span>
    <span class="sr-only">Siguiente</span>
  </a>
</div>
</div>
<script>
  $(document).ready(function(){
    // Activa Carrusel
    $("#slide_Carousel").carousel();
    // Habilita Evento click en botones (abajo) Aquí en esta seccion se crean la
    // programación con JQuery para activar el carrusel dinámico con los botones de
    // enlace a cada imagen (abajo imagen)
    $(".slide1").click(function(){
      $("#slide_Carousel").carousel(0);
    });
    $(".slide2").click(function(){
      $("#slide_Carousel").carousel(1);
    });
    $(".slide3").click(function(){
      $("#slide_Carousel").carousel(2);
    });
    $(".slide4").click(function(){
      $("#slide_Carousel").carousel(3);
    });

    // Habilita boton de desplazamiento anterior Aquí en esta seccion se crean la
    // programación para dar clic en el botón de ir a la imagen anterior (<)
    $(".left").click(function(){
      $("#slide_Carousel").carousel("prev");
    });
    // Habilita boton de desplazamiento siguiente Aquí en esta seccion se crean la
    // programación para dar clic en el botón de ir a la imagen siguiente (>)
    $(".right").click(function(){
      $("#slide_Carousel").carousel("next");
    });
  });
</script>

```

```
</body>  
</html>
```

## RESULTADO FINAL



### 3.2 JQuery

jQuery fue creado en 2006 por John Resig. Fue diseñado para resolver las incompatibilidades de los diferentes navegadores y para gestionar de una manera más sencilla el DOM de HTML, el manejo de eventos, las animaciones y Ajax. Ahora veremos varios ejemplos de código de JQuery.

jQuery se fundamenta en la programación orientada a objetos de javascript, en la creación de nuevos objetos, a estos mismos se les establecen propiedades y métodos brindando una mejor alternativa de programar.

La sintaxis de jQuery está definida para seleccionar elementos HTML y realizar acciones en los elementos correspondientes.

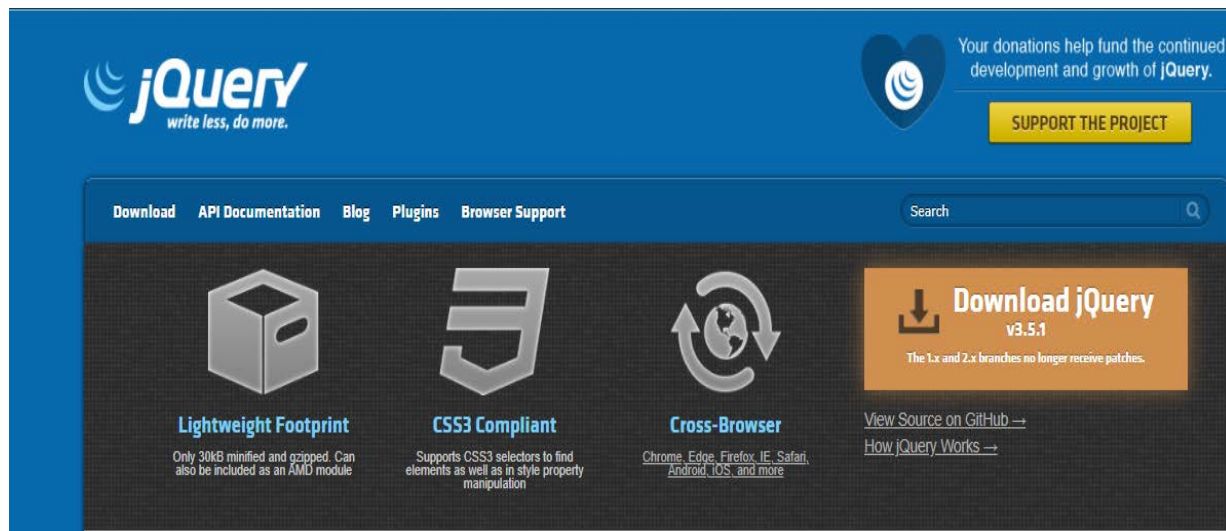
*El principal objeto utilizado es `jQuery()`, también se puede reemplazarlo por medio del símbolo: `$()`.*

La sintaxis básica es: **`$ ( selector ). acción ()`**

Un signo \$ para definir / acceder a jQuery

Un ( selector ) para "consultar (o buscar)" determinados elementos HTML

Una acción jQuery () que se ejecutará en los elementos



*Recuerda que: Debes de copiar la ubicación del link de enlace con el archivo de JQuery.js al principio del documento HTML*

Estructura básica de JQuery		
Instrucción JQuery	Instrucción Javascript	Función
<code>\$(document).ready(function( { // instrucc. javascript y jQuery//</code>	<code>window.onload=function () { ... }</code>	<i>Realiza función al cargar página totalmente.</i>

});		
<b>Jquery Elementos HTML</b>		
Element.text();	Element.textContent();	<i>Asignar un texto a un elemento HTML</i>
Var texto.html()	Var texto=element.innerText;	<i>Obtener un texto de un elemento con &lt;&gt;...&lt;/&gt;HTML</i>
Var element.html("<p>Este es JQuery</p>")	Var element.innerHTML("<p>Este es JQuery</p>")	<i>Asignar un contenido de un elemento HTML</i>
Var dato=element.html()	Var dato=element.innerHTML;	<i>Obtener un contenido de un elemento HTML</i>
<b>Jquery estilos CSS</b>		
Element.hide();	Element.style.display="none"	<i>Ocultar elemento HTML</i>
Element.show();	Element.style.display="";	<i>Mostrar elemento HTML</i>
Element.css("font-size","35px"); Element.css("text-align","center");	Element.style.fontSize="35px"; Element.style.textAlign="center";	<i>Cambiar tamaño de Fuente de elemento y centrar textos</i>
<b>Jquery HTML DOM</b>		
\$("#identificador").remove();	element.parentNode.removeChild(element);	<i>Eliminar elemento HTML</i>

Jquery Metodo CLASS		
.addClass()	Ejemplo: \$("#fondo").addClass("bg-azul")	<i>Método que agrega el atributo Class a un elemento HTML</i>
.removeClass()	Ejemplo: \$("#fondo").removeClass("bg-azul")	<i>Método que elimina el atributo Class a un elemento HTML</i>
.toggleClass()	Ejemplo: \$("#fondo").toggleClass("bg-verde")	<i>Metodo que agrega el atributo Class sino la tiene y a su vez la elimina si ya la posee previamente</i>
.hasClass()	Ejemplo: If (\$("#fondo").hasClass("show")){ // Instrucciones JS y JQuery// }	<i>Metodo que se utiliza para preguntar mediante la estructura condicional if si un elemento tiene el atributo Class indicado previamente</i>

### 3.2.1 EVENTO DOCUMENT READY()

En los anteriores ejemplos todos los métodos jQuery están incluidos de un evento de alistamientos de documentos:

```
$(document).ready(function){
```



```
// Metodos jQuery se coloquen acá ...
```

```
});
```

Esto es para evitar que se ejecute cualquier código jQuery antes de que el documento termine de cargarse (cuando esté listo completamente).

### 3.2.2 METODO TEXT()

#### EJEMPLO 1 DE JQUERY

```
<!DOCTYPE html>
<html>
<head>
<script src="js/Jquery.js"></script>
</head>
<body>
```

```
<h1>Ejemplo 1 - Programación Web</h1>
```

```
<h2 id="txt1">Javascript</h2>
```

```
<h2 id="txt2">jQuery</h2>
```

```
<script>
```

// En este script aplicamos JQuery utilizando el método text, observa que con una función a pesar de estar escrito Javascript en el id="txt1" se accede a ese elemento y se coloca un nuevo nodo de texto "HTML-CSS".

```
$(document).ready(function() {
```

```
    var texto = $("#txt1");
```

```
    texto.text("HTML-CSS");
```

```
});
```

```
</script>
```

```
</body>
```

```
</html>
```



### 3.2.3 METODO HTML()

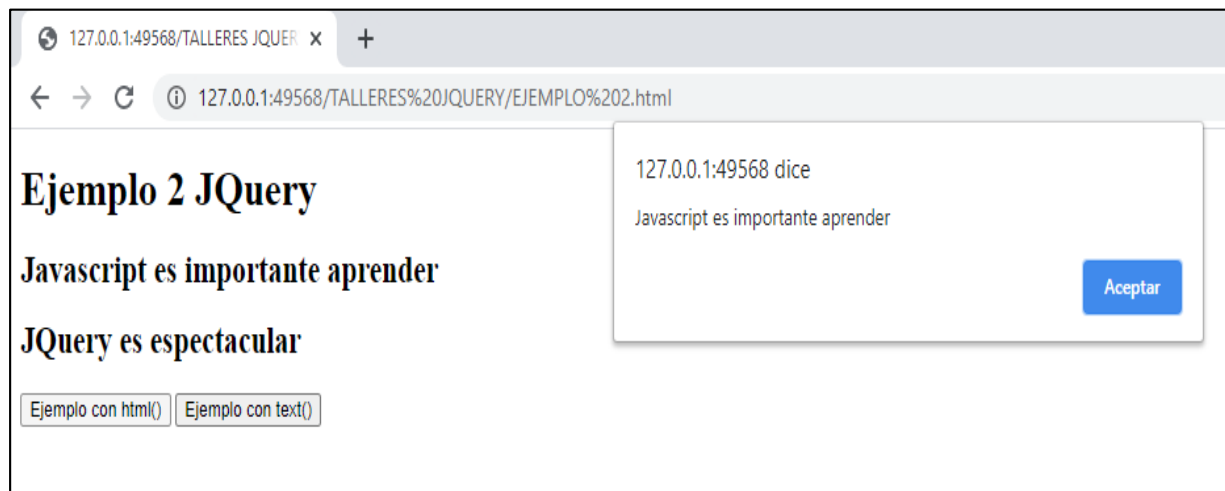
#### EJEMPLO 2 DE JQUERY

```
<!DOCTYPE html>
<html>
<head>
<script src="js/Jquery.js"></script>
</head>
<body>
<h1>Ejemplo 2 JQuery</h1>
<h2 id="txt1">Javascript es importante aprender</h2>
<h2 id="txt2">JQuery es espectacular estudiar</h2>
  <input type="button" id="exp1" value="Ejemplo con html()">
  <input type="button" id="exp2" value="Ejemplo con text()">
</body>
</html>
<script>
$(document).ready(function() {
  // En este script aplicamos JQuery utilizando los métodos html() y text(), observa que se
  // programa una función con el evento click en los dos botones para que muestre mediante una
  // ventana dialogo los nodos de textos de los dos <id>
  //Ejemplo con método html//
  $("#exp1").click(function(){
    alert($("#txt1").html());
  });
});
```

```

    });
    //Ejemplo con método text()//
    $("#exp2").click(function(){
        alert($("#txt2").text());
    });
});
</script>
</body>
</html>

```



### 3.2.4 EVENTO CLICK y METODO CSS()

#### EJEMPLO 3 DE JQUERY

```

<!DOCTYPE html>
<html>
<head>
<script src="js/Jquery.js"></script>
</head>
<body>
<h1>Ejemplo 3 de Programación JQuery</h1>
<div id="contenedor"></div>
<br>
<input type="button" value="cambiar color de fondo" id="btn1" />

```

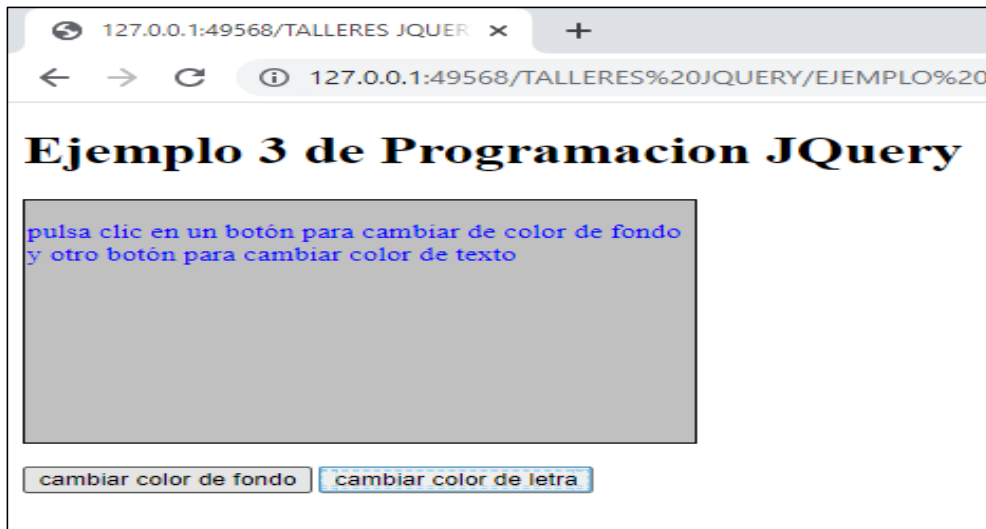
```

<input type="button" value="cambiar color de letra" id="btn2" />
<style>
  #contenedor {
    width: 350px; height: 200px; border: 2px solid black; display: block;
  }
</style>
<script>
// En este script aplicamos JQuery utilizando el método css() y el evento click(), observa que se crean dos botones con el evento click : con una función un boton para aplicar con método css el color de fondo gris y con otra función otro boton que al dar clic cambia a color azul al texto definido en contenedor

$(document).ready(function(){
  $("#contenedor").html("<p>pulsa clic en un botón para cambiar de color de fondo y otro botón para cambiar color de texto</p>");
  // se aplica estilos al contenedor de color de fondo
  $("#btn1").click(function(){
    $("#contenedor").css("background-color", "silver");
  });
  // se aplica estilos al contenedor de color de letra
  $("#btn2").click(function(){
    $("#contenedor").css("color", "blue");
  });
});
</script>
</body>
</html>

```

## RESULTADO FINAL



### 3.2.5 METODO CLASS()

EJEMPLO CON METODO .TOGGLECLASS()

```
<!DOCTYPE html>
<html>
<head>
<script src="js/Jquery.js"></script>
</head>
<body>
<h1>Ejemplo 4 de Programación JQuery</h1>

<div id="contenedor">
<span id="text1">Pulsa botón para ver metodo .toggleClass</span></div>
<br>
<input type="button" value="Clic acá para cambiar tamaño, color y tipo de fuente" id="btn1"
/>

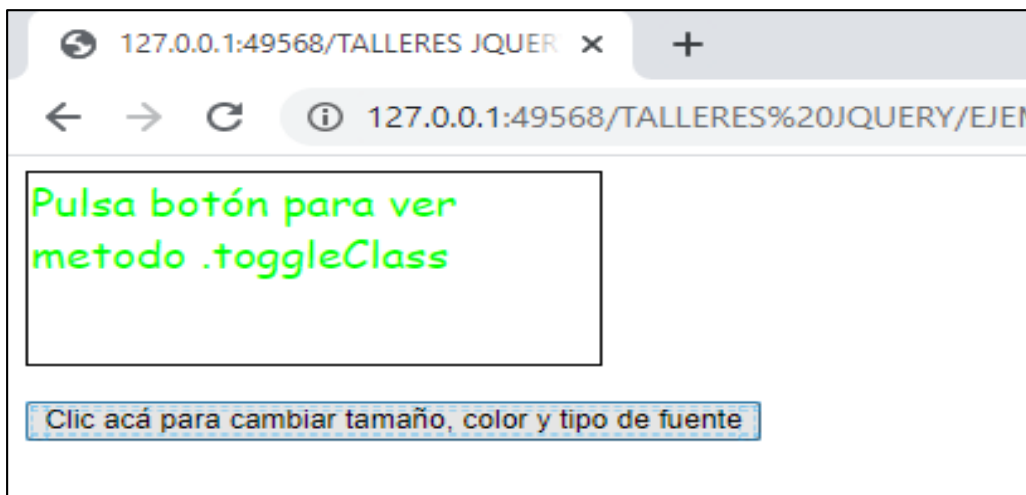
<style>
#contenedor {
width: 250px; height: 100px; border: 2px solid black; display: block;
}
.estilos-class{
font-family: Comic Sans Ms;
```

```

    font-size: 20px;
    color:lime;
  }
</style>
<script>
// En este script aplicamos JQuery utilizando el método .toggleClass() y el evento click(),
observa que se crean un botón con el evento click : con una función para aplicar estilos css
de tipo de letra: Comic Sans Ms , tamaño de letra a 20 pixeles y color de letra verde lima al
texto definido en contenedor al pulsar clic en botón.
$(document).ready(function(){
    $("#btn1").click(function(){
        $("#text1").toggleClass("estilos-class");
    });
});
</script>
</body>
</html>

```

## RESULTADO FINAL



### 3.2.6 EVENTOS TOGGLE()

```
<!DOCTYPE html>
<html>
<head>
<script src="js/Jquery.js"></script>
</head>
<body>
<h1>Ejemplo 5 de Programación JQuery</h1>

<div id="contenedor">
<span id="text1">Pulsa botón para ver eventos toggle</span>
  <span id="text2">Aprende a programar en Javascript</span>
  <span id="text3">y también aprende a programar en JQuery</span>
</div>
<br>
<input type="button" value="Clic acá para efecto fundido.." id="btn1" />
<input type="button" value="Clic acá para efecto ver/ocultar.." id="btn2" />
<style>
  #contenedor {
    width: 250px; height: 100px; border: 2px solid black; display: block;
  }
</style>

<script>
// En este script aplicamos Jquery utilizando el evento toggle y el evento click(), observa que
se crean dos botones con el evento click : Un boton con el metodo fadeToggle con una
duración de 2000 ms la cual presenta una animación de fundido y otro boton con el metodo
slideToggle con una duración de 2000 ms la cual presenta una animación de deslizamiento.
$(document).ready(function(){
//Evento fundido
  $("#btn1").click(function() {
    $("#text1").fadeToggle(2000);
  });
//Evento deslizamiento
  $("#btn2").click(function(){
```

```

        $("#text1").slideToggle(2000);
    });
});
</script>
</body>
</html>

```

## RESULTADO FINAL



### 3.3 JQuery con @media Queries y Side Bar Left

Ahora vamos a aplicar programación JQuery en nuestra página web para dispositivos móviles -responsive- aplicando el estilo CSS con @media queries con Side bar left, para miremos cuales son:



## EJEMPLO:

```
<!DOCTYPE html>
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1">
  <script src="js/Jquery.js"></script>
</head>
```

```
<style>
body {
  font-family: Verdana;
```

/\* En este ejemplo se aplica estilos css al menu principal position fixed para fijar a la ventana, z-index:10 para colocar encima del texto, desplazamiento top:0 y left:0 para que salga desde la izquierda, color de fondo black (negro)con overflow-x:hidden esconder parte sobrante de contenedor con una transición de 0.5 seg y un espacio arriba de 61pixeles \*/

```
.mainNav {
  height: 100%;
  width: 0;
  position: fixed;
  z-index: 10;
  top: 0;
  left: 0;
  background-color: black;
  overflow-x: hidden;
  transition: 0.5s;
  padding-top: 61px;
}
```

/\* En este ejemplo se aplica estilos css a los hiperenlaces <a> con espacios en pixeles, con color blanco, con display block (bloque), tamaño de fuente 2 pixeles y una transicion de 0.4 seg. Se aplica estilos con Pseudoclasas hover para que al pasar mouse sobre opciones de menu se cambia a color azul claro, Se aplica estilos al icono de cerrar (X) con position absolute con desplazamiento a la derecha de 25 pixeles, con tamaño 36 pixeles y margen izquierdo 150 pixeles \*/

```
.mainNav a {
  padding: 8px 8px 8px 25px;
  text-decoration: none;
```

```
font-size: 22px;
color: white;
display: block;
transition: 0.4s ease;
}
```

```
.mainNav a:hover {
  color: lightblue;
}
```

```
.mainNav .closebtn {
  position: absolute;
  top: 0;
  right: 25px;
  font-size: 36px;
  margin-left: 150px;
}
```

/\* En este ejemplo se aplica estilos css diseñando un @media queries hasta un máximo de ventana de 800 pixeles el cual se coloca un color de fondo en degradado azul opaco con un grado de opacidad \*/

```
@media screen and (max-width: 800px) {
  body{
    background-color:rgba(16, 150, 155, 0.7);
  }
}
```

```
</style>
```

```
<body id="body">
```

/\* En este ejemplo se aplica estilos insertando una imagen de Misc symbols &times es un icono X cerrar \*/

```
<div id="mainNav" class="mainNav">
  <a href="#" class="closebtn" id="close">&times;</a>
  <a href="#">Inicio</a>
  <a href="#">Productos</a>
  <a href="#">Servicios</a>
  <a href="#">Contacto</a>
</div>
```

```

<h2>Side Bar Left de navegación</h2>
<p>Este diseño de Side Bar permite colocar la seccion de opciones -nav- saliendo desde el
lado izquierdo del navegador con solo dar Click en el icono de hamburguesa.</p>
/* En este ejemplo se aplica programación Jquery al icono de hamburguesa (&#9776) el cual
al darle clic aparece del lado izquierdo con un ancho de 200pixeles y color de fondo oscuro
atenuado con un grado de opacidad
<div style="font-size:30px;cursor:pointer" id="open">&#9776; Click acá</div>

<script>
    $(document).ready(function(){
// Evento clic en icono hamburguesa, en este ejemplo se aplica programación
Jquery al icono de abrir (&#9776;) el cual al darle clic aparece hacia el lado
izquierdo arriba con un ancho de 200 pixeles y negro pálido el fondo página.
        $("#open").click(function() {
            $("#mainNav").css("width","200px");
            $("#body").css("backgroundColor","rgba(0,0,0,0.4)");

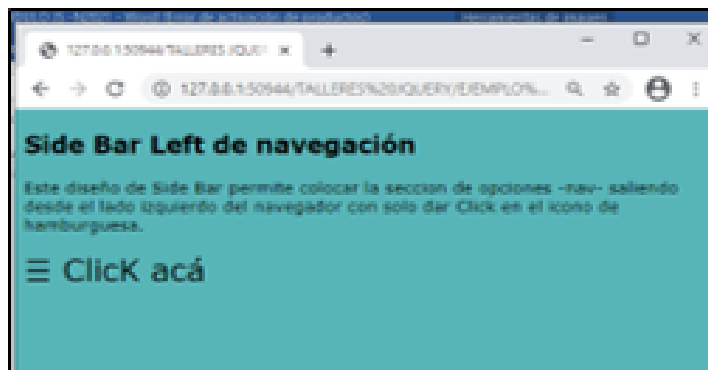
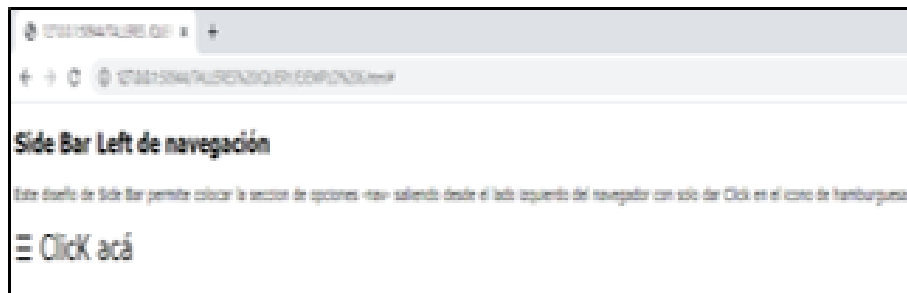
        });
// Evento clic en icono X cerrar, en este ejemplo se aplica programación Jquery
al icono de cerrar (&times;) el cual al darle clic desaparece hacia el lado izquierdo
con 0 pixeles y color de fondo blanco y recargando página totalmente.
        $("#close").click(function(){
            $("#mainNav").css("width","0px");
            $("#body").css("backgroundColor","white");
            window.location.reload();

        });
    });
</script>

</body>
</html>

```

## RESULTADO FINAL



Ventana con max-width:800px



Activar  
Ve a Conf

Ahora vamos a aplicar programación JQuery en nuestra página web de proyecto en clase para dispositivos móviles CON @media queries –responsive- debemos de hacer varios cambios y agregar elementos HTML, para miremos cuales son:

### En HTML:

```
<script src="js/jquery.js"></script><!-- cambio -->
</head>
<body>
<!-- seccion cabecera -->
<header><center>
><!-- cambio -->
<h1> CREACIONES QUE LINDURA</h1>
</center></header>
<!-- seccion menú -->
<nav id="menu"><!-- cambio -->
  <div class="openmenu"> <i class="fas fa-bars"></i></div>
  <!-- Estudiante debes de aplicar estos cambios al documento original INDEX.html observa las
  líneas de color negro y tienen un comentario con la palabra cambio -->
  <ul class="mainmenu"><!-- cambio -->
    <i class="fas fa-home"></i>
    <li><a href="#">INICIO</a></li>
    <i class="fas fa-cart-arrow-down"></i>
    <li><a href="html/Galeria.html" target="_blank">PRODUCTOS</a></li>
    <i class="fas fa-charging-station"></i>
    <li><a href="#">SERVICIOS</a></li>
    <i class="fab fa-wpforms"></i>
    <li><a href="html/contactenos.html">CONTACTENOS</a></li>
    <div class="closemenu"> <i class="fas fa-times"></i></div><!-- cambio -->
```

### En CSS:

```
/* Se aplica estilos al menu principal con display flex(flexible), al icono de cerrar con atributo
pointer que al pasar mouse sobre icono aparece una mano, y no se muestra a la vista.*/
nav .mainmenu{
  display: flex;
}
```

```

/* estilo encadenado con menu */
nav .mainmenu .closemenu{
    display: none;
    cursor:pointer;
    font-size: 2rem;
}
/*icono hamburguesa-cambio, Se aplica estilos a icono de abrir (hamburguesa)
con atributo pointer que al pasar mouse sobre icono aparece una mano, y no se
muestra a la vista.  */
nav .openmenu{
    display: none;
    cursor:pointer;
    font-size: 2rem;
    margin: 20px;
}
/*icono hamburguesa fontawesome, Se aplica estilos a icono de abrir (hamburguesa) con
propiedad tamaño de fuente 32 pixeles y color blanco*/
.fa-bars{
    font-size: 32px;
    color: white;
}
/* Se aplica estilos a icono de cerrar (X) con propiedad tamaño de fuente 32 pixeles y color
blanco */
.fa-times{
    font-size: 32px;
    color: white;
}

/* @media queries aplicada a disp. Moviles máximo 800px */
@media(max-width:800px){
body{
    background-color: rgba(0,0,0,0.4);
    background-image: url(../img/ejemplo-logotipo.gif);
    background-repeat: no-repeat;
}
nav .openmenu{
    display: block;

```

```
position: absolute;
top: 10px;
right: 10px;
z-index: 99;
cursor: pointer;
}
```

*/\* Se aplica estilos con metodo @media queries con los siguientes: a la página web total se diseña color de fondo gris atenuado con opacidad y se inserta imagen de logotipo y que no se repita.*

*Al icono de abrir (hamburguesa) se establece position: absoluta con márgenes derecho y propiedad z-index para que se coloca encima del texto de página con efecto de pointer (puntero mouse).*

*Se aplica al menu principal ancho y alto 100%, con position fixed para que se ajuste a la ventana, con desplazamiento top y left 0 para que aparezca desde la izquierda, con sobreposicion al texto de página, con display flex con direccion de columna y centrado tanto horizontal y verticalmente, con transicion y color fondo.\*/*

```
nav .mainmenu{
height: 100%;
width: 100%;
position: fixed;
top: 0;
left: 0;
overflow-x: hidden;
z-index: 10;
display: flex;
flex-direction: column;
justify-content: center;
align-items: center;
background-color: rgba(0,0,0,0.4);
transition: all 4.8s ease;
display: none;
}
```

*/\* estilo encadenado con menu, Se aplica estilos a icono de cerrar (X) con display block, con atributo pointer que al pasar mouse sobre icono aparece una mano, y se muestra a la vista al momento de aparecer menu, con position: absoluta con margen izquierda 10 pixeles*

```

*/
nav .mainmenu .closemenu{
display: block;
position: absolute;
top: 10px;
right: 10px;
margin-left: 500px;
cursor: pointer;
}
nav .mainmenu li{
padding: 12px;
}
/* Se aplica estilos a las secciones header (encabezado), section, aside, article (cuerpo
página), footer (pie página) con display none para que no se vea a la vista*/
header, section, aside, article, footer{
display: none;
}
}

```

## En JQuery:

```

<script>
// En este ejemplo se aplica programación JQuery, se crean variables para almacenar los
<id> de manejo de menu principal
$(document).ready(function(){
    var openmenu = $(".openmenu");
    var closemenu = $(".closemenu");
    var mainmenu = $(".mainmenu");

    // Evento clic en icono hamburguesa, En este ejemplo se aplica programación JQuery, al
    icono hamburguesa con evento clic para que aparezca menu principal y desaparece icono de
    hamburguesa
    openmenu.click(function() {
        mainmenu.css("display", "flex");
        openmenu.css("display", "none");
    });

    // Evento clic en icono X cerrar, En este ejemplo se aplica programación JQuery, al icono X

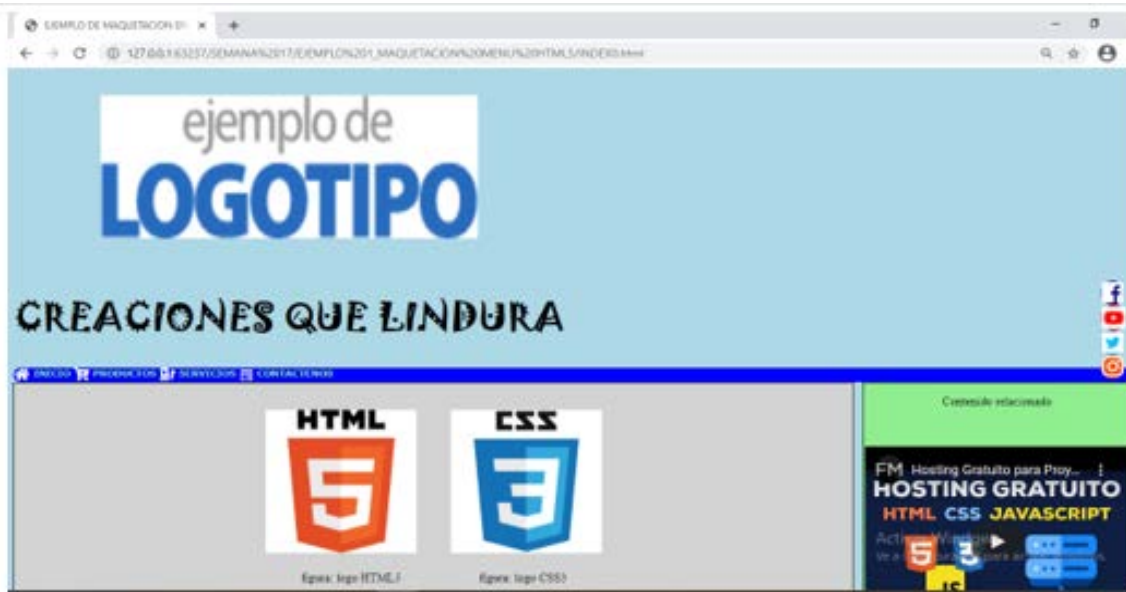
```



*cerrar con evento clic para que desaparezca menu principal y recargue página con metodo reload*

```
closemenu.click(function() {  
    mainmenu.css("width","0");  
    openmenu.css("display","block");  
    window.location.reload();  
});  
});  
</script>
```

## RESULTADO FINAL



### *Cambio de Ancho ventana a 800px*



## **FDEH (Formador dice y estudiante hace)**

Estudiante recuerda guardar todos los formularios, hojas de estilo y otros documentos creados en las carpetas correspondientes de PROYECTO EN CLASE y para reafirmar conocimientos adquiridos debes de realizar proyecto final del trimestre aplicando todos los temas aprendidos en clase, por favor pedir instrucciones a su formador.

**Valoración de Evidencias:**

## GLOSARIO

**Algoritmo:** Serie de instrucciones o reglas finitas y definidas que se deben seguir pasos secuenciales para resolver un problema determinado.

**APIs:** Interfaces de Programación de Aplicaciones (APIs) construidas dentro de los navegadores que permiten hacer cualquier cosa, desde crear contenido HTML y establecer estilos CSS, hasta capturar y manipular un video desde la webcam, o generar gráficos 3D y sonidos de ejemplo

**Desarrollador:** es una persona que crea programas en diferentes lenguajes de programación.

**Instrucciones:** Son un conjunto de líneas de código de programación, lo que permitirá dar órdenes a la PC para gestionar un programa.

**ECMA International:** (European Computer Manufacturers Association, la Asociación Europea de Normalización de hoy los sistemas de información y comunicación)

ES 5 (publicada en diciembre de 2009), que es la versión más reciente liberada; ES 6, que se encuentra actualmente en fase de diseño.

**JavaScript:** es un lenguaje script u orientado a documento. Se ejecuta en un navegador, se une a frameworks o librerías como la programación en el servidor con Node.js o Angular.js

**jQuery:** Frameworks de tercera generación y librerías, como jQuery, que se aplica en cualquier página web HTML facilitando su programación y publicación web.

**Lenguaje hipertexto:** lenguaje de etiquetas o tags en donde el navegador las analiza y procesa en el momento que deben ser ejecutadas. se puede embeber en una página web HTML

**Lenguaje de programación:** Conjunto de instrucciones que son interpretadas por un computador para realizar operaciones y procesos diversos. Por ejemplo el lenguaje de programación JavaScript.

**Programación orientada a objetos:** un lenguaje de programación orientado a objetos es un lenguaje que contiene elementos, llamados objetos y diferentes tienen características específicas y formas de uso diferente llamadas clases, métodos y propiedades.

**Sintaxis:** se define como el conjunto de reglas que deben seguirse al escribir el código fuente de los programas para la correcta ejecución del lenguaje de programación.

## CIBERGRAFÍA

FERRER JORGE, GARCÍA VÍCTOR, GARCÍA RODRIGO. Curso completo de HTML.[en línea].Disponible en Internet: <http://freek.jorgeferrer.com>.

EGUÍLUZ PÉREZ JAVIER. Introducción a JavaScript.[en línea].Disponible en Internet: [www.librosweb.es](http://www.librosweb.es).

RODRÍGUEZ JOSE ANTONIO. Manual De JavaScript.[en línea].Disponible en Internet: [www.internetmania.net](http://www.internetmania.net).

ALVAREZ MIGUEL ANGEL, GUTIERREZ MANU. Manual De JavaScript.[en línea].Disponible en Internet: [desarrolloweb.com/manuales/manual-javascript.html](http://desarrolloweb.com/manuales/manual-javascript.html).

LOPEZ IVAN DAVID. Categoria Javascript[en línea]. Disponible en Internet: <https://byspel.com/>

<https://lenguajejs.com/javascript/>

<https://www.w3schools.com/js/default.asp>

<https://www.tutorialesprogramacionya.com/javascriptya>

<https://code.tutsplus.com/es/tutorials/>

<https://aprende-web.net/javascript/>

<https://es.stackoverflow.com/tags>

<https://yeisondaza.com/>

<https://3con14.biz/js/>