## 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.

## 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);

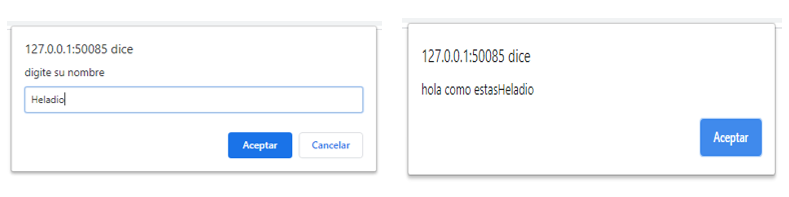
**}**

var nombre=prompt( “digite su nombre”');

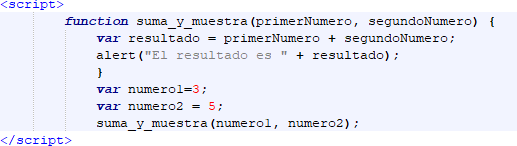
**bienvenida(nombre);**

</script>

RESULTADO FINAL



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

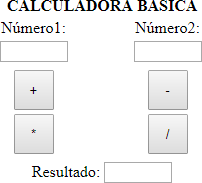


Vamos a listar algunas características de los argumentos de una función:

* 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)

## 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 la 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 recomienda.

Las funciones externas igualmente son un método muy sencillo pero que en aplicaciones complejas presentan alguno que 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 type="text/javascript">

var pestana;

**function abrir(){**

pestana=window.open("https://www.incap.edu.co")

**}**

**function cerrar(){**

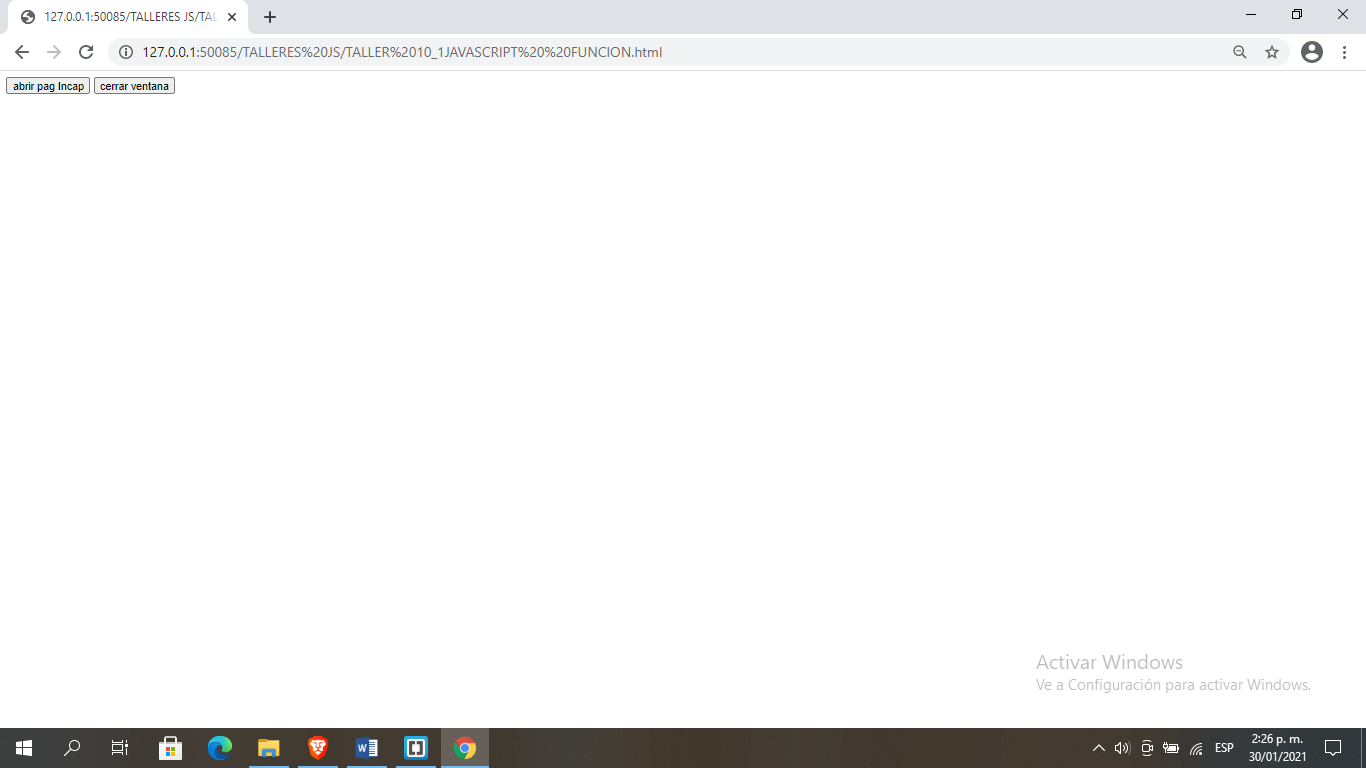
pestana.close();

**}**

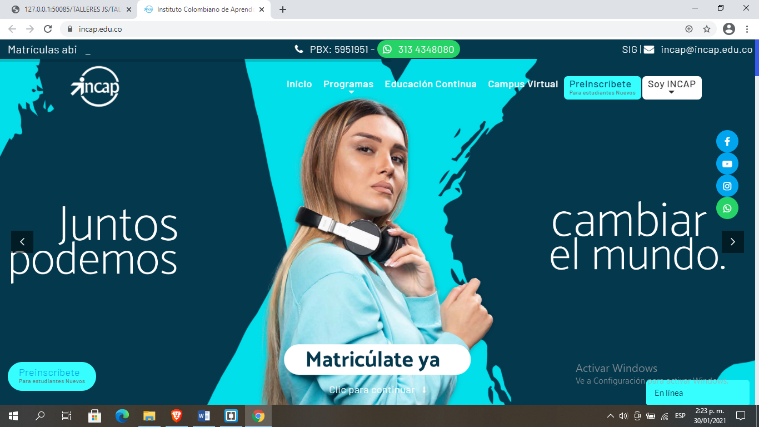
</script>

**RESULTADO FINAL**

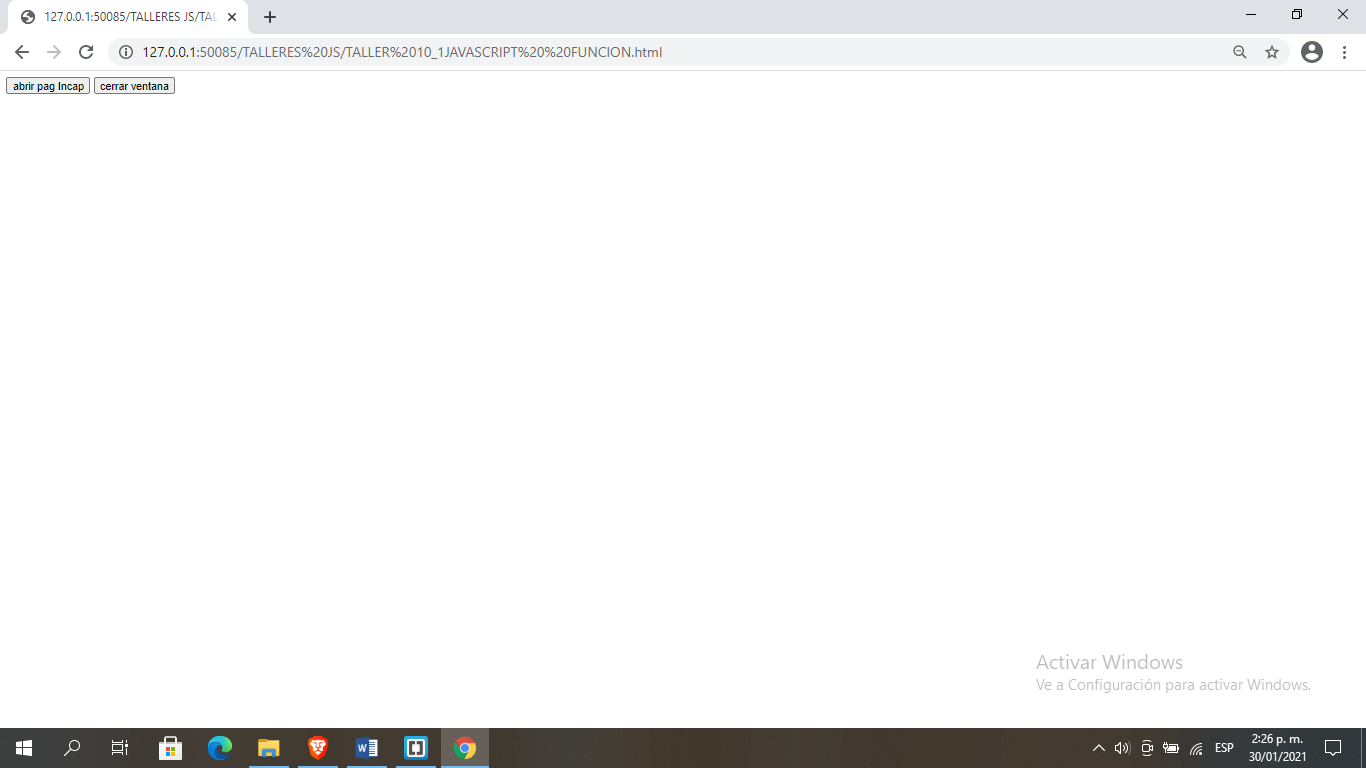
**Vista final, dar clic en abrir pág. Web Incap**



**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>

## **Barra menú lateral Colapsada**

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

**EJEMPLO:**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type"content="text/html" charset="UTF-8">

<title>EJEMPLO DE BARRA LATERAL COLAPSADA EN HTML 5</title>

<link rel="stylesheet" type="text/css" href="css/mestilo1nuevo.css">

<link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.8.1/css/all.css" integrity="sha384-50oBUHEmvpQ+1lW4y57PTFmhCaXp0ML5d60M1M7uH2+nqUivzIebhndOJK28anvf" crossorigin="anonymous">

</head>

<body id="content">

<!-- seccion cabecera -->

<header>

<h1> BARRA MENU LATERAL COLAPSADA</h1>

</header>

<!-- seccion menú -->

<nav id="menu">

<div id="content-menu">

<!-- seccion de icono de hamburguesa para barra menú -->

<button class="openbtn" onclick="**openMenu**()">&#9776;MENU SIDEBAR</button>

<!-- seccion de redes sociales en barra menú -->

<div id="**redes**">

<a href="https://www.facebook.com"><i class="fab fa-facebook-square"></i></a>

<a href="https://www.youtube.com"><i class="fab fa-youtube-square"></i></a>

<a href="https://www.twitter.com"><i class="fab fa-twitter-square"></i></a>

<a href="https://www.instagram.com"><i class="fab fa-instagram"></i></a>

</div>

<!-- seccion de icono lupa para buscador para barra menú -->

<div class="**buscarbox**">

<i class="fas fa-search"></i>

<input type="text" placeholder="Buscar aquí....">

</div>

<!-- seccion de icono de usuario para registrar cliente -->

<div id="**reguser**">

<a href="#""><i class="fas fa-user">&nbsp;Registrar</i></a>

</div>

</div>

<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 opernMenu para abrir la barra lateral colapsada y oscurecer fondo de pagina web

**function openMenu() {**

document.getElementById("mainMenu").style.width = "200px";

document.getElementById("mainMenu").style.height = "800px";

document.getElementById("content").style.backgroundColor="rgba(0,0,0,0.5)";

document.getElementById("menu").style.backgroundColor="rgba(0,0,0,0.5)";

**}**

// -- Sección de programacion de funcion para el botón closeMenu para cerrar la barra lateral colapsada y recargar pagina web.

**function closeMenu() {**

document.getElementById("mainMenu").style.width = "0";

window.location.reload();

**}**

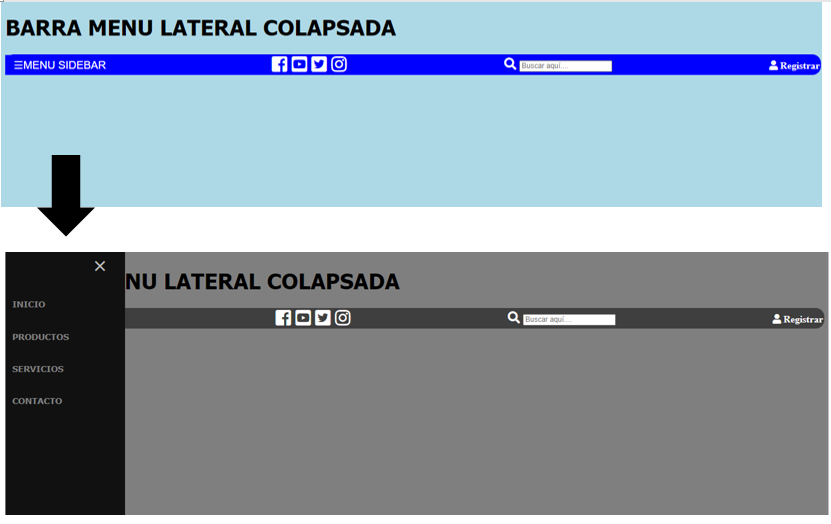
</script>

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

</body>

</html>

**RESULTADO FINAL**



* 1. **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, métodos y eventos de Javascript.

**EJEMPLO:**

En HTML

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type"content="text/html" charset="UTF-8">

<tittle></tittle>

<link rel="stylesheet" type="text/css" href="css/Productos.css">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">

</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>

<img src="img/portatiles1.jpg" width="100" height="100">

<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>

<img src="img/portatiles2.jpg" width="100" height="100">

<h3>$2,199.000</h3>

<button id="button-addo" onclick="**carro**()">Añadir carrito</button></center>

</div>

</div>

<!-- Sección Modal: 1era imagen-->

<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" onclick="**cerrar**();">Seguir comprando</button>

</div>

</div>

<!-- Sección Modal: 2da imagen-->

<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" onclick="**cerrarv2**();">Seguir comprando</button>

</div>

</div>

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

<script src="https://stackpath.bootstrapcdn.com/bootstrap/3.4.1/js/bootstrap.min.js" integrity="sha384-aJ21OjlMXNL5UyIl/XNwTMqvzeRMZH2w8c5cRVpzpU8Y5bApTppSuUkhZXN0VxHd" crossorigin="anonymous"></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 cerrar(){**

document.getElementById("win-modal").style.display="none";

}

//\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**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";

}

}

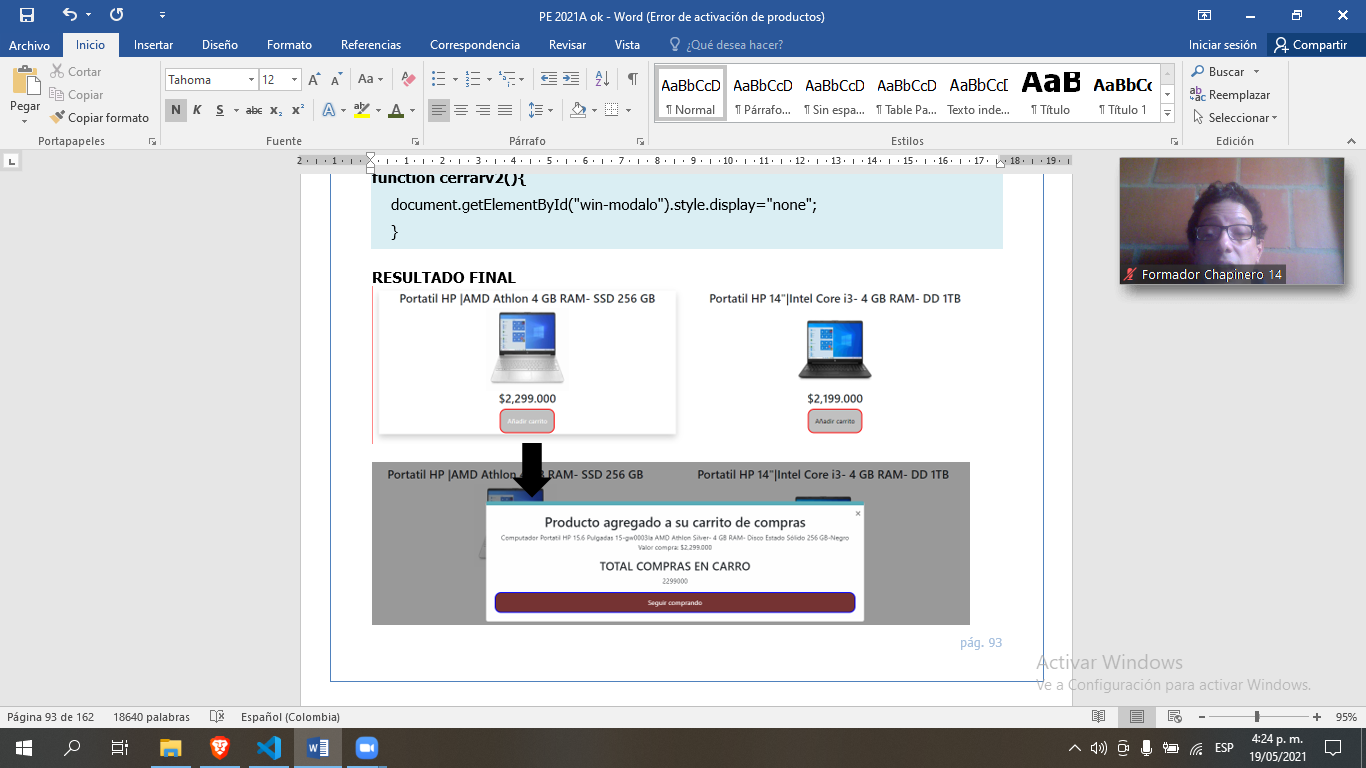
}

**function cerrarv2(){**

document.getElementById("win-modalo").style.display="none";

}

**RESULTADO FINAL**



* 1. **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');**

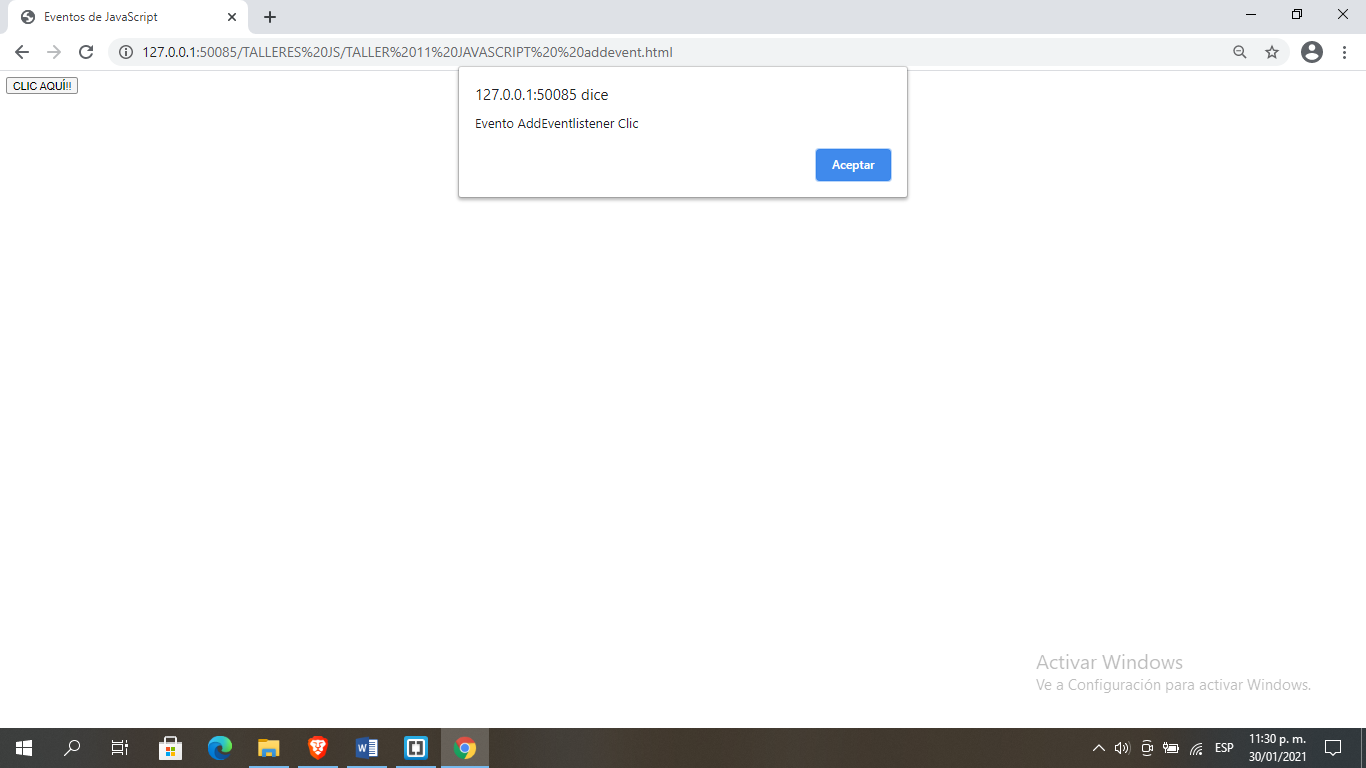
**boton.addEventListener('click',function()=>{**

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');**

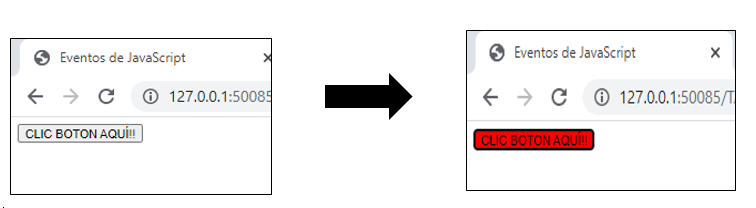
**boton.addEventListener('click',function(){**

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(){**

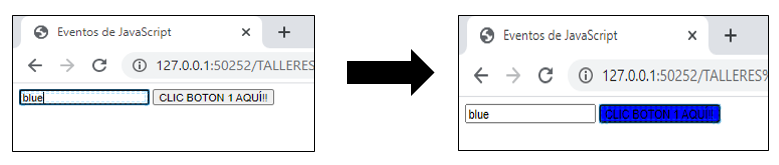
var color=document.getElementById("color").value;

boton.style.backgroundColor=color;

})

</script>

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



* 1. **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';

}

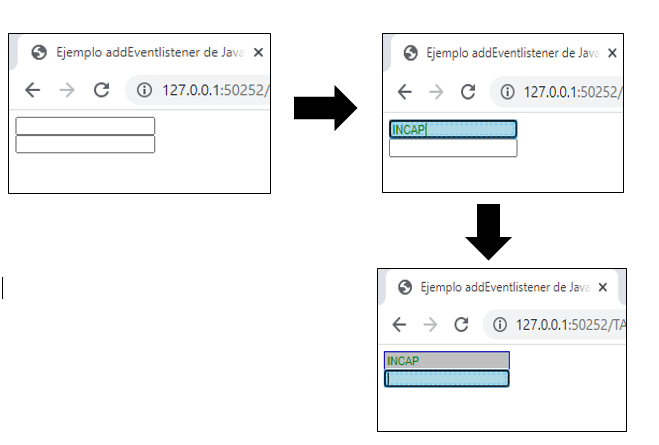
function salir\_input2()

{

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

}

</script>



* 1. **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.

* + 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 | function() { return this.categoria + " " + this.marca +" "+this.color |

**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*;**

**};**

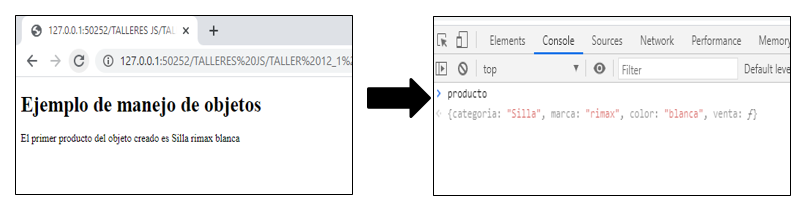
document.getElementById("mensaje").innerHTML =

"El primer producto del objeto creado es… " + producto.venta();

</script>

</body>

</html>



* + 1. **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 () {

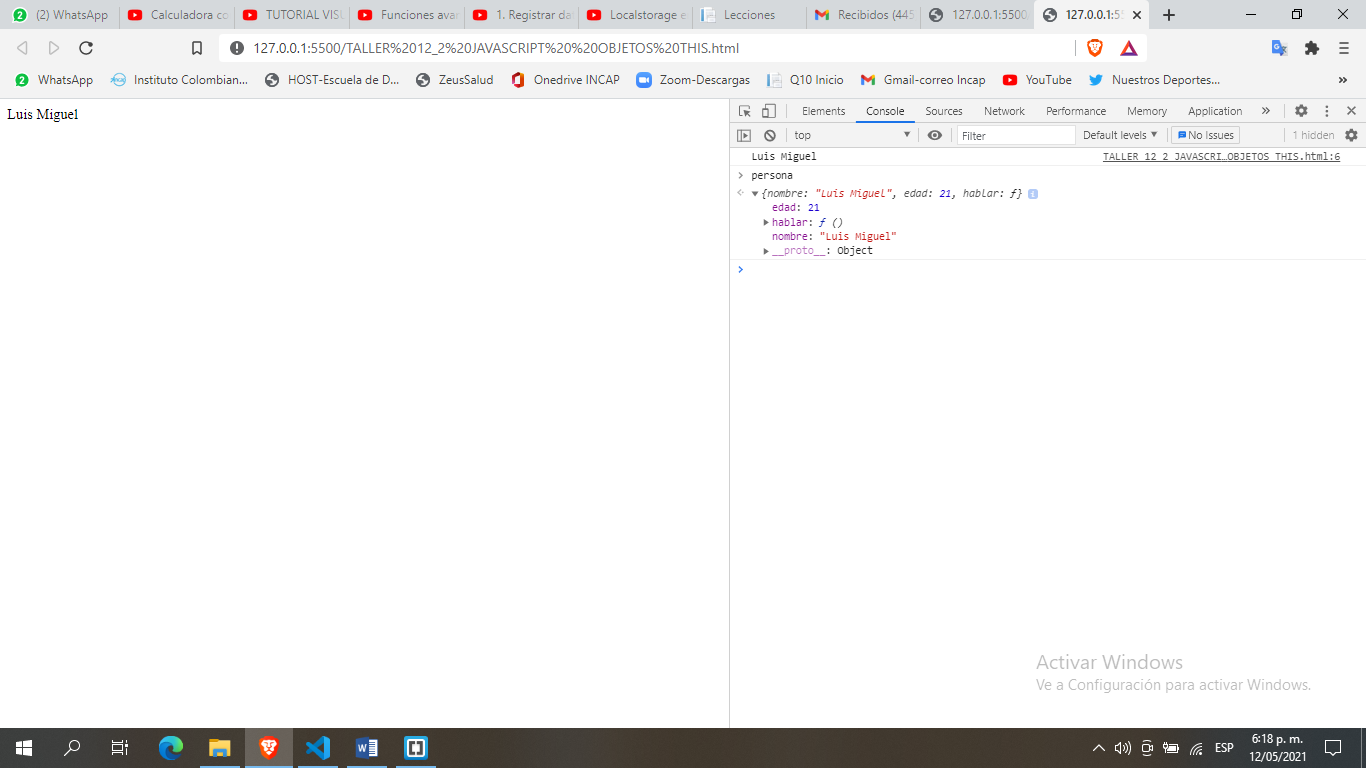
console.log(this.nombre);

}

};

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.

## 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.

## 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.

* + 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;



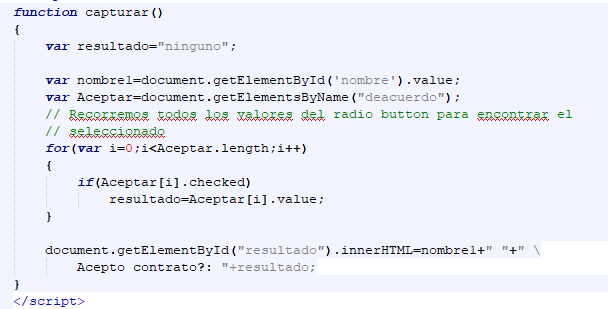
El código anterior crea el formulario que permite capturar los datos.

* + 1. **Botones de opción (radiobutton):**

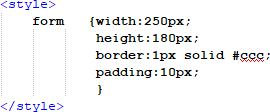
En este objeto solo se selecciona una única opción y en muchos casos queremos saber cuál de ellos fue seleccionado; para ello recurrimos a la propiedad checked que devuelve verdadero si el botón de opción fue seleccionado y falso en caso contrario.

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.



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.

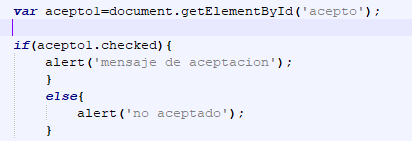


El código anterior da un formato sencillo al formulario, utilizando reglas CSS.

* + 1. **Casillas de verificación (checkbox):**

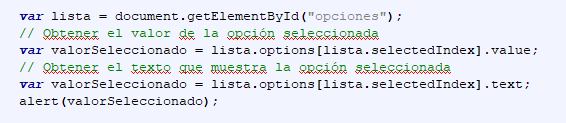
Las casillas de verificación tienen una forma similar de capturar su información a la de los radiobutton, con la diferencia que se pueden seleccionar varias casillas a la vez y cada checkbox se debe comprobar de manera independiente y no en conjunto como los radiobutton. Esta característica ya la conocen 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:***



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

* + 1. **Cuadros combinados (select):**



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.

## 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.

* + 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; }**

* + 1. 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;**

**}**

* + 1. **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;**

**}**

* + 1. **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;**

**}**

* + 1. **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;**

**}**

* + 1. **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;**

**}**

* + 1. **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;**

**}**

**FDH (Formador y dice)**

* + 1. **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="&#9889 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="&#9977 Ingrese nombre y apellido" required autofocus autocomplete="off" pattern="[A-Za-z ]{2,30}">**

**<br>**

**<input type="email" id="email" placeholder="&#9746 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="&#9742 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="&#9757 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 envio desde 0-->*

**<button class="btn\_enviar" type="submit" >ENVIAR</button>**

**<button class="btn\_limpiar" type="reset" >CANCELAR</button>**

**</div>**

**</form>**

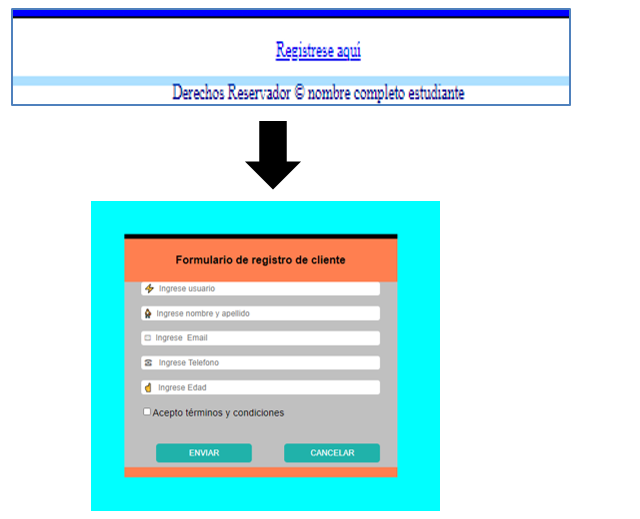
</div>

</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**



* 1. **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.

* + 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. |

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

Caracteres especiales:

|  |  |
| --- | --- |
| **Carácter especial** | **Función** |
| \d | Concordar con un dígito |
| \s | Concordar con un espacio en blanco |
| \w | Concordar con una letra o digito |

**Cuantificadores**:

|  |  |
| --- | --- |
| **cuantificador** | **Función** |
| string+ | Concordancia puede aparecer mínimo 1 o más vez |
| string\* | Concordancia puede aparecer 0 o más veces. |
| string? | Concordancia puede aparecer o no puede aparecer. |
| {n} | Longitud de n caracteres. |
| {n,} | Longitud de n o más caracteres (sin límite) |
| {n1,n2} | Longitud de caracteres de n1 hasta n2 |

## **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+1lW4y57PTFmhCaXp0ML5d60M1M7uH2+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="">

<!-- seccion de control input para ingresar usuario -->

<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>

<!-- seccion de control input para ingresar email de usuario -->

<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>

<!-- seccion de control input para ingresar contraseña de usuario -->

<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>

<!-- seccion de control input para ingresar verificación de contraseña de usuario -->

<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");

// seccion para programar bloqueo de clic en formulario y ejecuta función validar inputs -->

form.addEventListener('submit', e =>{

e.**preventDefault**();

validainputs();

});

// seccion para programar función validar inputs del formulario -->

function validainputs() {

const usuarioval= usuario.value.trim();

const emailval= email.value.trim();

const passwordval= password.value.trim();

const password2val= password2.value.trim();

// seccion para programar si ocurrre un error en el input de ingreso de usuario si es vacío se activa la función de error en los inputs en este caso el del usuario -->

if (usuarioval ==="") {

ocErrorform(usuario, "no puede usuario estar vacío");

}else{

exitosform(usuario);

}

// seccion para programar si ocurrre un error en el input de ingreso de email de usuario si es vacío se activa la función de error en los inputs en este caso el del email de 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);

}

// seccion para programar si ocurrre un error en el input de ingreso de password si es vacío se activa la función de error en los inputs en este caso el password del usuario -->

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);

}

// seccion para programar si ocurrre un error en el input de ingreso de password si es vacío se activa la función de error en los inputs en este caso el password del usuario y luego valida si las contraseñas son iguales -->

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);

}

}

// seccion para programar función ocErrorform al momento de ocurrir un error en los inputs del formulario activa los estilos de error definidos-->

function ocErrorform (input, message){

const formControl=input.parentElement;

const small=formControl.querySelector("small");

formControl.className="form-control error";

small.innerText =message;

}

// seccion para programar función exitosform para colocar los estilos de exitos definidos en el formulario -->

function exitosform (input){

const formControl=input.parentElement;

formControl.className="form-control success";

}

// seccion para validar mediante una expresión regular correspondiente al email del usuario del formulario si está bien ingresado al input-->

function valEmail(email){

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

}

// seccion para validar mediante una expresión regular correspondiente a la contraseña del usuario del formulario si está bien ingresada al input-->

function valPassword(password){

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

}

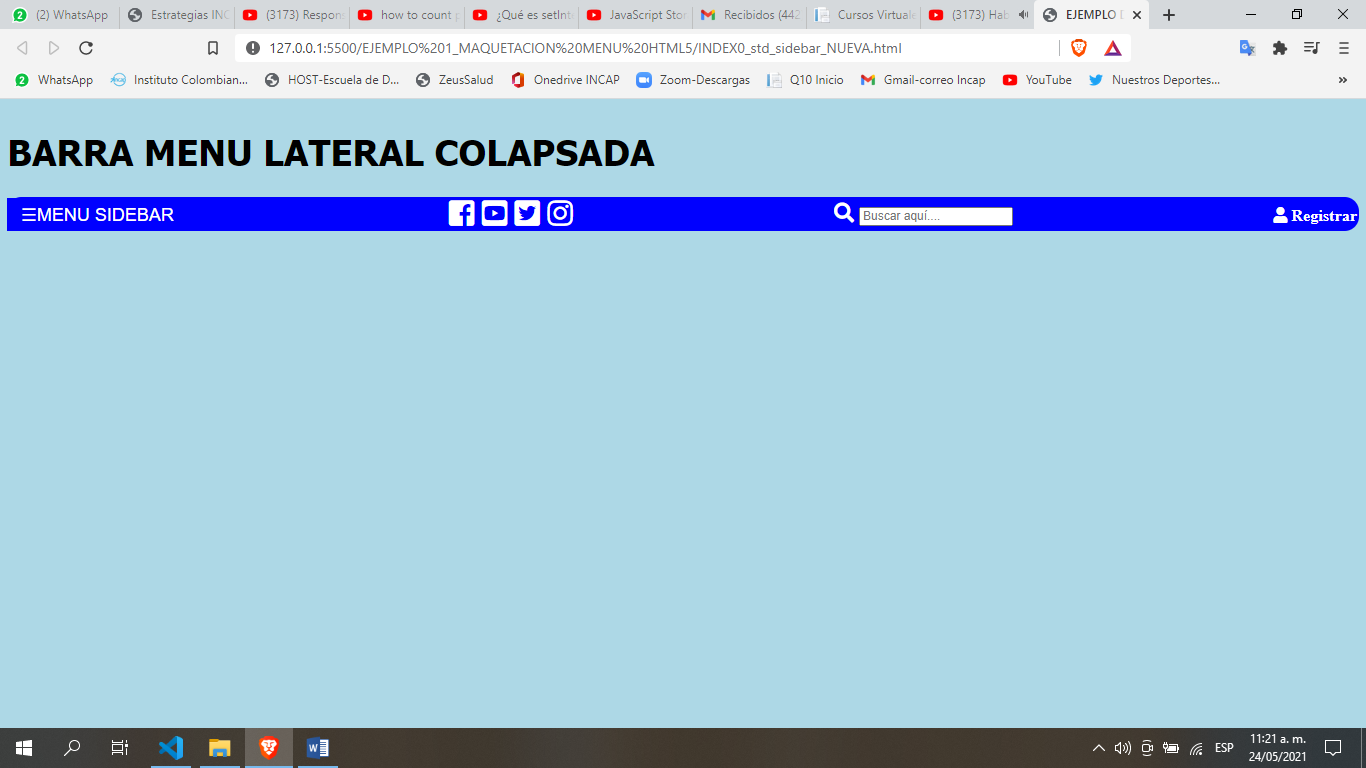
**RESULTADO FINAL**



* 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 javascripten la subcarpeta de js con el nombre **validarformavanz.js**

***Recuerda que:***  *enlazar el archivo de documento html con el documento HTML* ***INDEX.html*** *(proyecto anterior en clase) en el icono de usuario de fontawesome localizado en la barra menú con la siguiente línea:*

<a href=" **html/formulario\_validavanza.html**"><i class="fas fa-user">&nbsp;Registrar</i></a>



**VALIDACION DE FORMULARIO DE CONTACTO**

Ahora vamos a crear una validación de datos al formulario de registro cliente de nuestra página web vista en el trimestre pasado en HTLM 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,expcorreo;

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

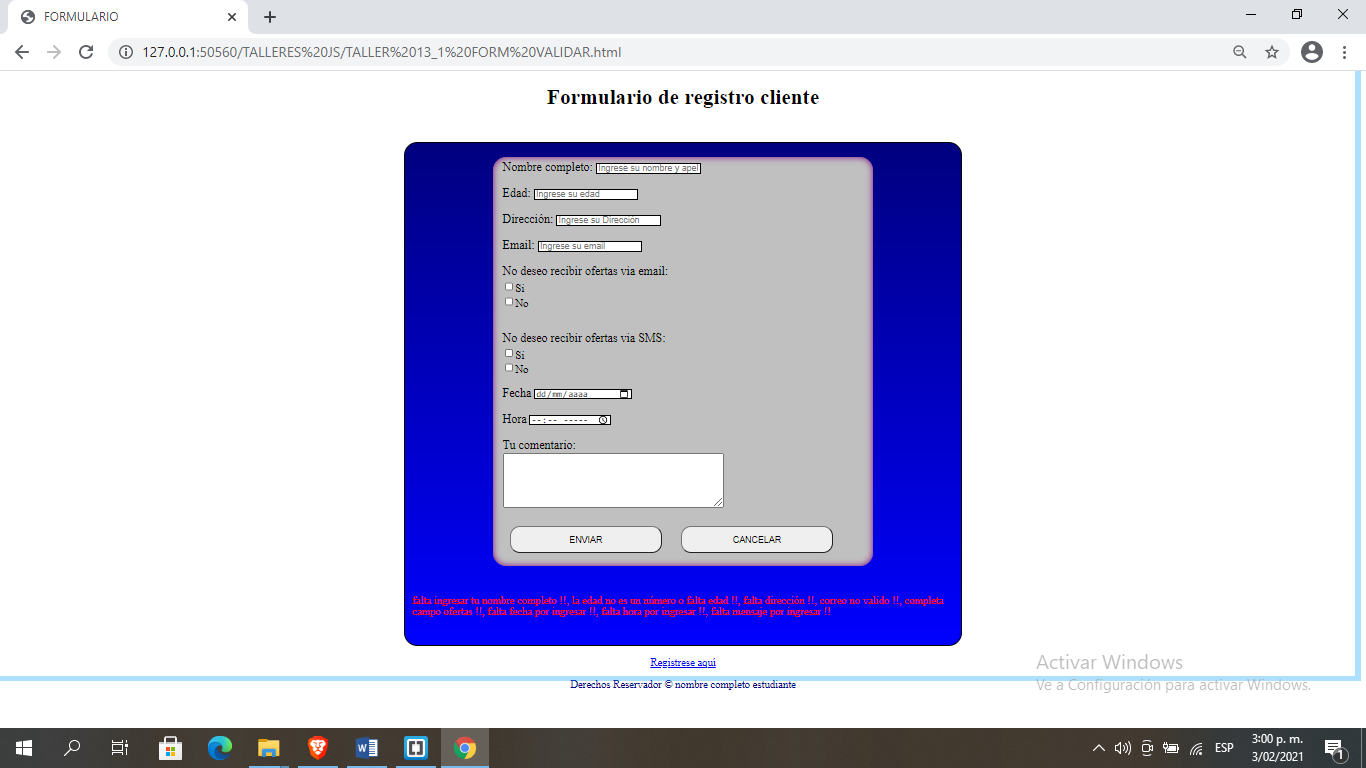
}

</script>

</body>

</html>

**RESULTADO FINAL**



**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**

1. **FRAMEWORK (LIBRERÍAS) EN JS**

* 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.

* 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 Acá 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 Acá 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">

<img src="img/foto5.jpg" alt="diseñoweb1" width="460" height="345">

<div class="carousel-caption">

<h3>Diseño web 1</h3>

<p>Diseño web es espectacular.</p>

</div>

</div>

<!-- imagenes slides Acá se colocan las imágenes que tendrá el carrusel tipo slide con un titulo y un texto referente. -->

<div class="item">

<img src="img/foto1.jpg" alt="diseñoweb2" width="460" height="345">

<div class="carousel-caption">

<h3>Diseño web 2</h3>

<p>HTML estudia ya es importante.</p>

</div>

</div>

<!-- imagenes slides Acá se colocan las imágenes que tendrá el carrusel tipo slide con un titulo y un texto referente. -->

<div class="item">

<img src="img/foto2.jpg" alt="diseñoweb3" width="460" height="345">

<div class="carousel-caption">

<h3>Diseño web 3</h3>

<p>Javascript estudia ya es excelente.</p>

</div>

</div>

<div class="item">

<img src="img/foto3.jpg" alt="diseñoweb4" width="460" height="345">

<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 Acá 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) Acá 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 Acá 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 Acá 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");

});

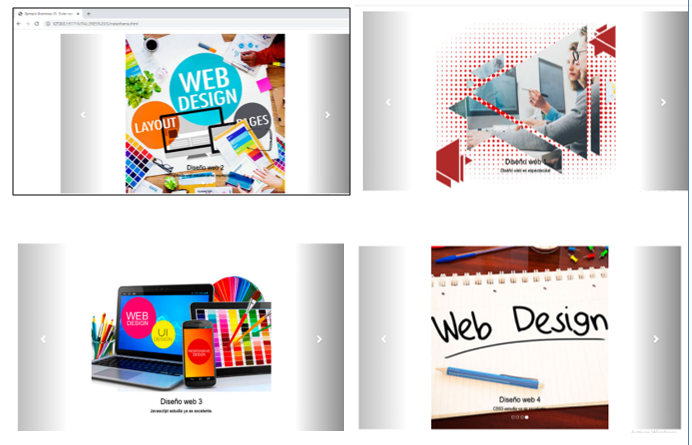
**});**

</script>

</body>

</html>

**RESULTADO FINAL**



* 1. ***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 codigo 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.

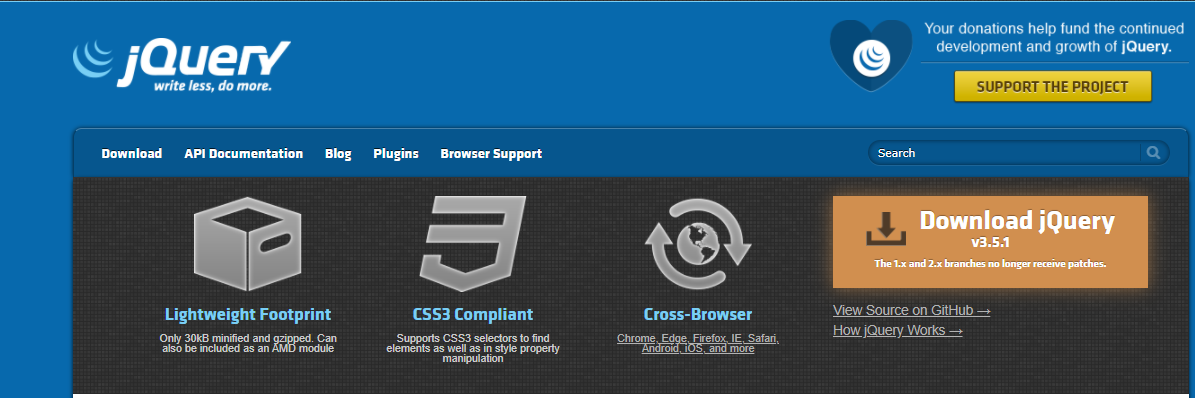
*El principal objeto utilizado es jQuery(), también se puede reemplazarlo por medio del símbolo: $().*

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** | |
| $(document).ready(function({  // instrucc. javascript y jQuery//  }); | window.onload=function () {  ... } | | *Realiza función al cargar página totalmente.* |
| **Jquery Elementos HTML** | | | |
| Element.text(); | Element.textcontent(); | | *Asignar un texto a un elemento HTML* |
| Var texto.html() | Var texto=element.innerText; | | *Obtener un texto de un elemento con <>…</>HTML* |
| Var element.html(“<p>Este es Jquery</p>”) | Var element.innerHTML(“<p>Este es Jquery</p>”) | | *Asignar un contenido de un elemento HTML* |
| Var dato=element.html() | Var dato=element.innerHTML; | | *Obtener un contenido de un elemento HTML* |
| **Jquery estilos CSS** | | | |
| Element.hide(); | Element.style.display=”none” | | *Ocultar elemento HTML* |
| Element.show(); | Element.style.display=””; | | *Mostrar elemento HTML* |
| Element.css(“font-size,”35px”);  Element.css(“text-align”,”center”); | Element.style.fontsize=”35px”;  Element.style.textalign=”center”; | | *Cambiar tamaño de Fuente de elemento y centrar textos* |
| **Jquery HTML DOM** | | | |
| $(“#identificador”).remove(); | element.parentNode.removeChild(element); | | *Eliminar elemento HTML* |
| **Jquery Metodo CLASS** | | | |
| .addClass() | Ejemplo:  $(“#fondo”).addClass(“bg-azul”) | | *Método que agrega el atributo Class a un elemento HTML* |
| .removeClass() | Ejemplo:  $(“#fondo”).removeClass(“bg-azul”) | | *Método que elimina el atributo Class a un elemento HTML* |
| .toggleClass() | Ejemplo:  $(“#fondo”).toggleClass(“bg-verde”) | | *Metodo que agrega el atributo Class sino la tiene y a su vez la elimina si ya la posee previamente* |
| .hasClass() | Ejemplo:  If ($(“#fondo”).hasClass(“show”)){  // Instrucciones JS y Jquery//  } | | *Metodo que se utiliza para preguntar mediante la estructura condicional if si un elemento tiene el atributo Class indicado previamente* |

* + 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 colocen acá ...

});

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

* + 1. **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");

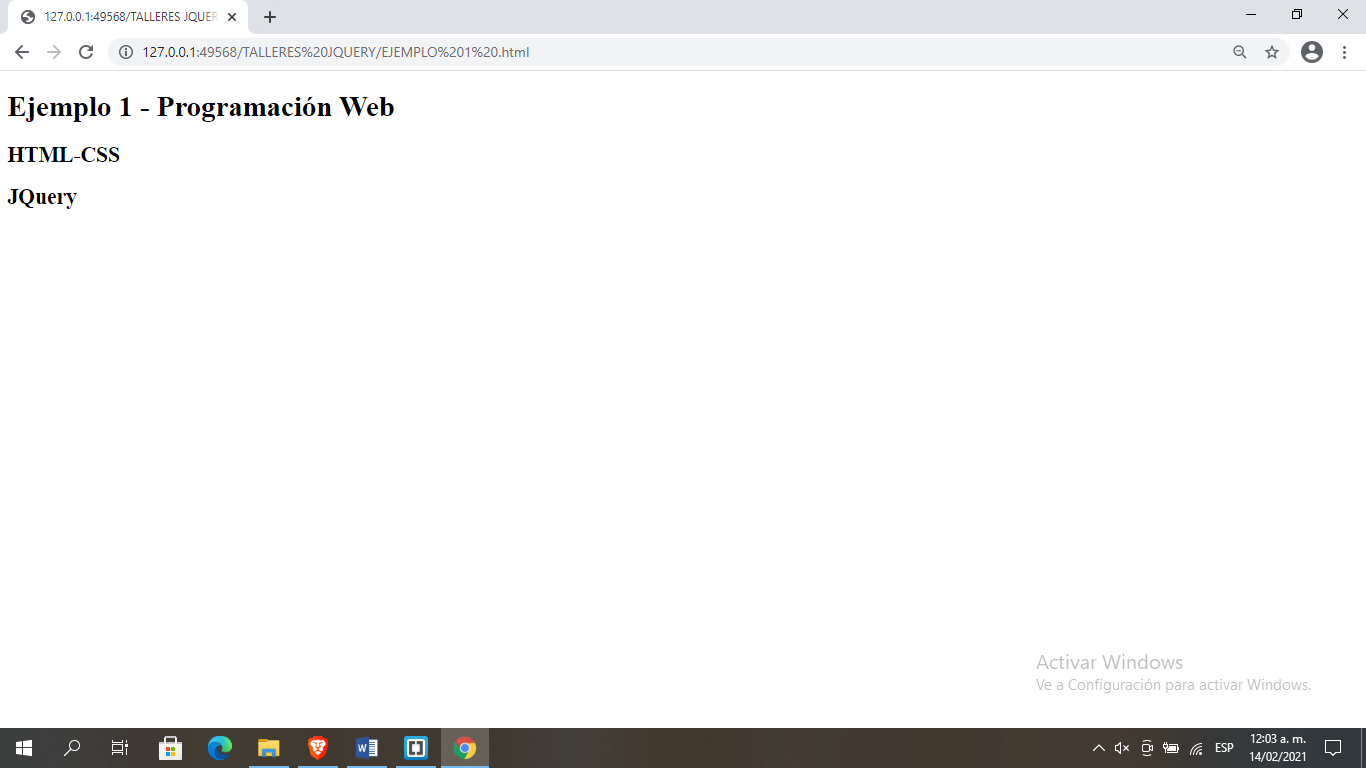
texto.text("HTML-CSS");

**});**

</script>

</body>

</html>



* + 1. **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()">

*<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());*

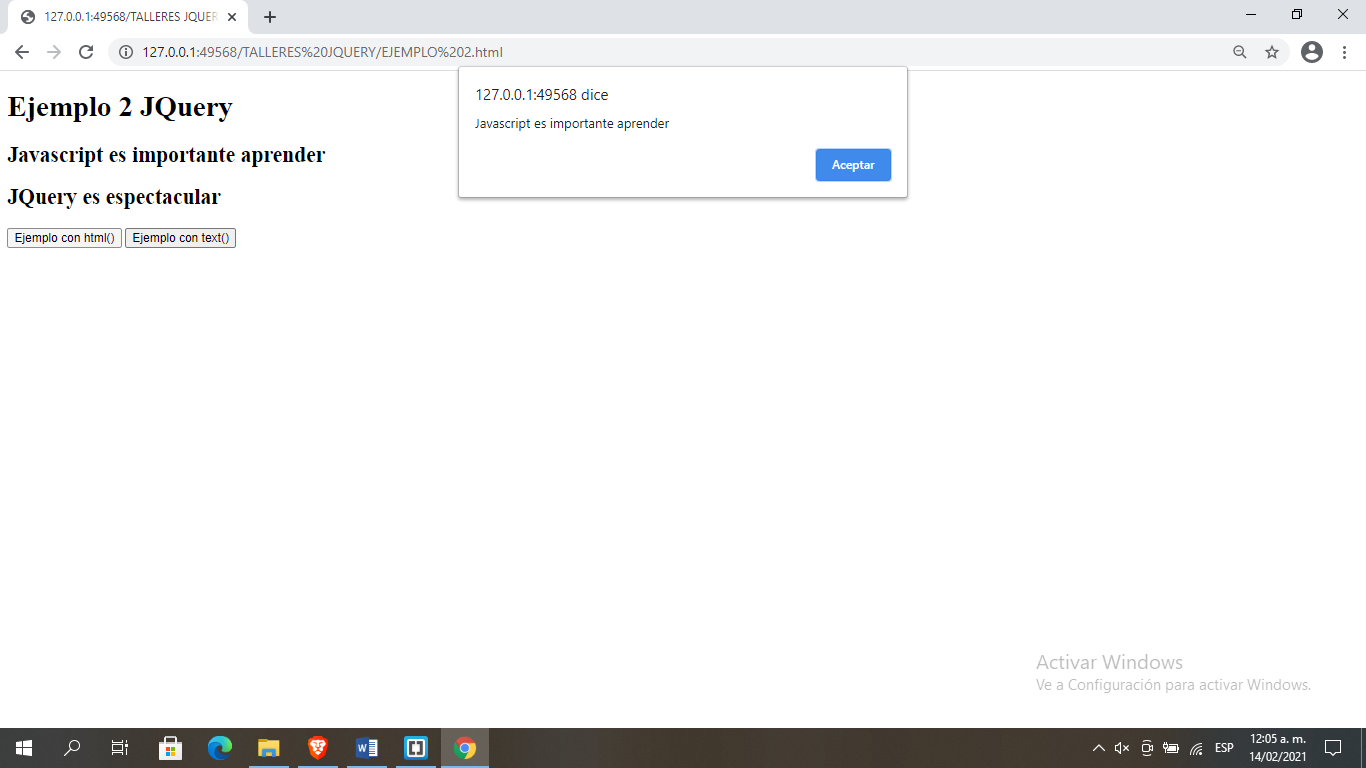
*});*

***});***

*</script>*

</body>

</html>



* + 1. **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");*

*});*

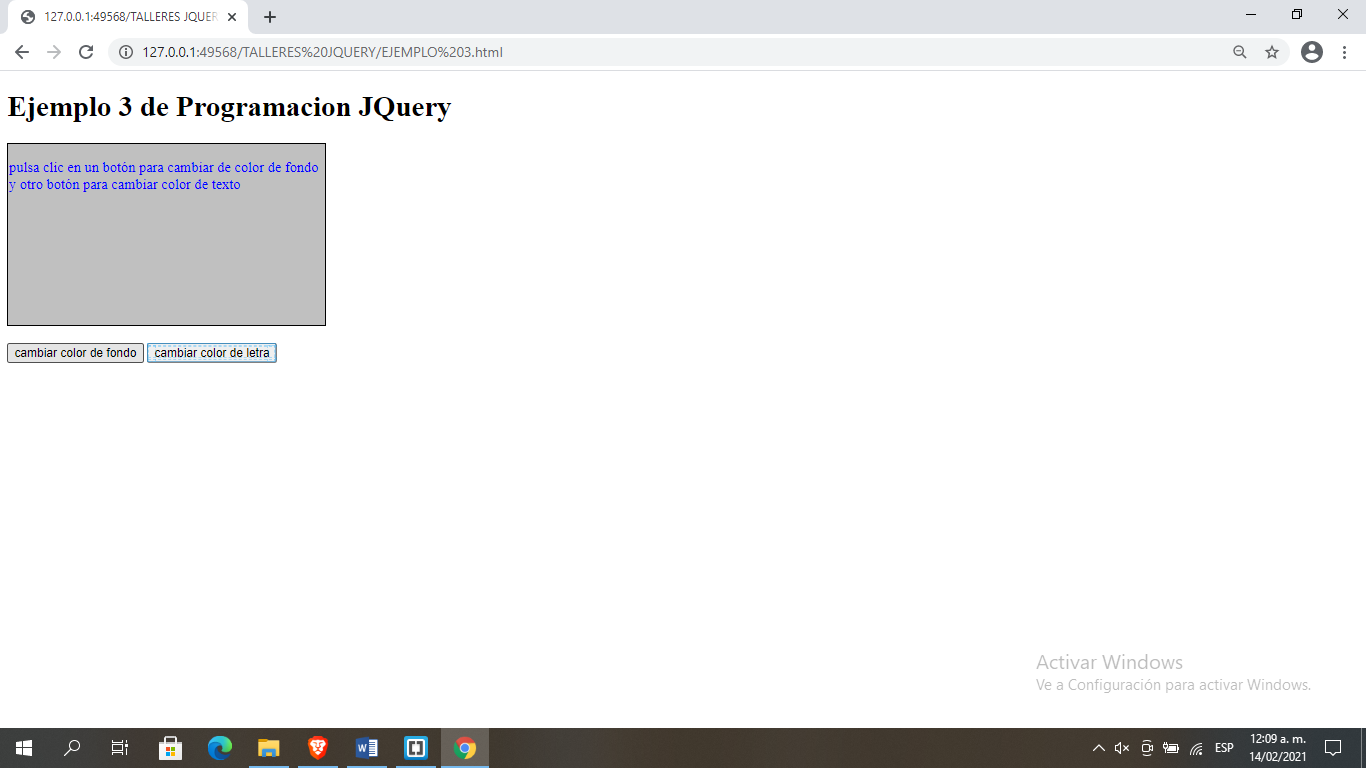
***});***

*</script>*

</body>

</html>

**RESULTADO FINAL**



* + 1. **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");

});

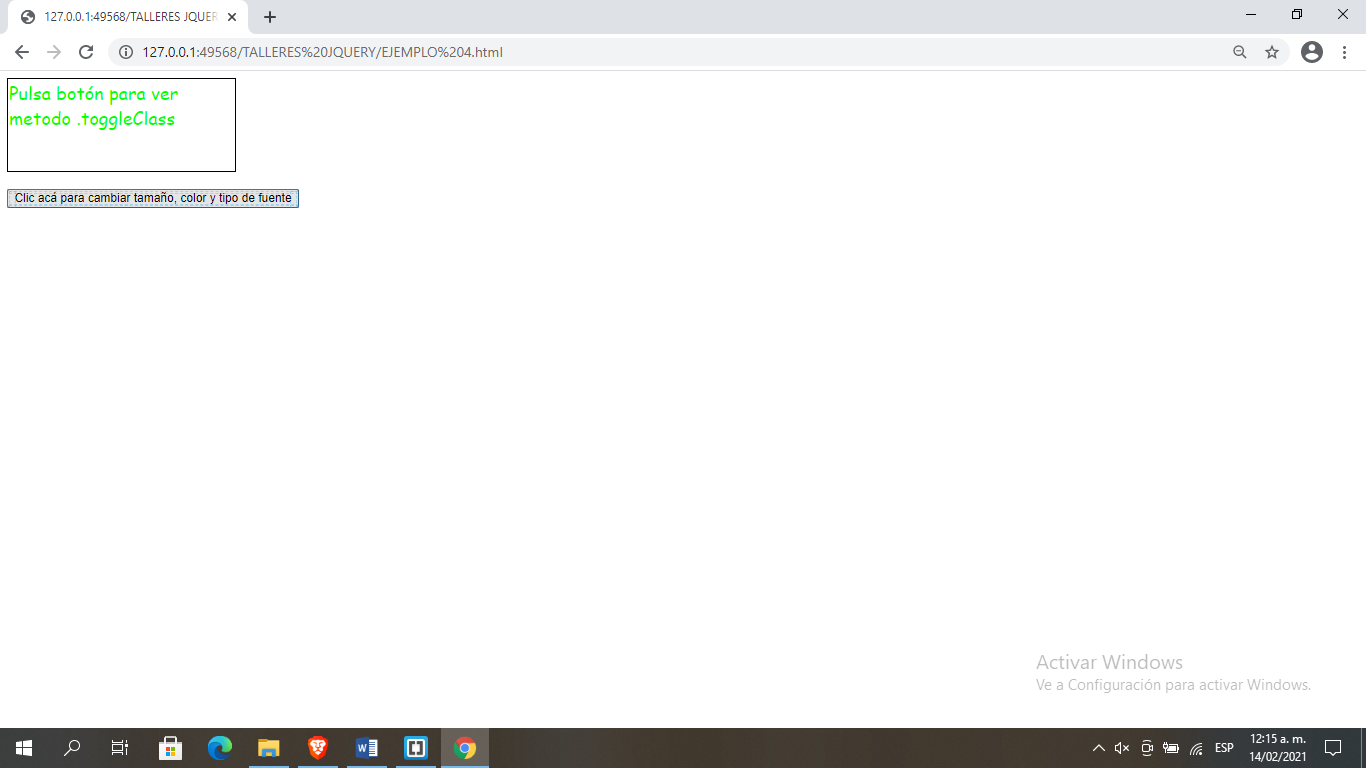
**});**

</script>

</body>

</html>

**RESULTADO FINAL**



* + 1. **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**



* 1. **Jquery con @media Queries y Side Bar Left**

Ahora vamos a aplicar programación JQuery en nuestra página web para dispositivos moviles –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 Pseudoclases 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();

});

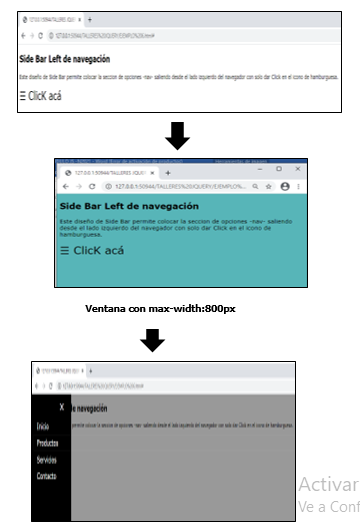
});

</script>

</body>

</html>

**RESULTADO FINAL**

****

Ahora vamos a aplicar programación JQuery en nuestra página web de proyecto en clase para dispositivos moviles 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>

*<img id="logo" src="img/ejemplo-logotipo.gif" width="500" height="200" alt="img-logo"/>><!-- 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**



**EJEMPLO DE FILTRO DE TABLAS**

Vamos a crear una tabla de datos en HTML y luego crear un botón input para filtrar los datos.

**EN HTML**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<h2>BUSCADOR DE CONTACTOS DE CLIENTES</h2>

<! - - Seccion donde insertamos una lupa junto a un control input para ingresar dato a buscar en tabla - ->

<img id="lupa" src="img/lupa.png" width="10px" height="10px">

<input type="text" id="datos" **onkeyup="Buscar()"** placeholder="Buscar por nombres de película..">

<! - - Seccion donde creamos y definimos la tabla de datos junto con los registros de clientes que tendrá almacenada. - ->

<table id="tablaDatos">

<tr class="header">

<th style="width:30%;">Nombre completo</th>

<th style="width:30%;">Direccion</th>

<th style="width:20%;">Edad</th>

<th style="width:20%;">Email</th>

</tr>

<tr>

<td>Pedro Castro</td>

<td>calle 48f 5-89 Bogotá</a></td>

<td>34</td>

<td>pcastro34@gmail.com</td>

</tr>

<tr>

<td>Maria Peréz</td>

<td>calle 4b 15-49 Bogotá</a></td>

<td>32</td>

<td>mperez32@gmail.com</td>

</tr>

<tr>

<td>Carmen Rios</td>

<td>Avenida 68 15-89 Bogotá</a></td>

<td>43</td>

<td>crios43@gmail.com</td>

</tr>

<tr>

<td>Adela Cuellar</td>

<td>calle 28c 56-189 Bogotá</a></td>

<td>24</td>

<td>acuellar24@gmail.com</td>

</tr>

<tr>

<td>Pepe Toro</td>

<td>cra. 14 45-38 Bogotá</a></td>

<td>29</td>

<td>ptoro29@gmail.com</td>

</tr>

</table>

**En CSS:**

\* {

box-sizing: border-box;

}

/\* Sección de estilos para control input de busqueda\*/

#datos {

width: 100%;

font-size: 16px;

padding: 12px 20px 12px 40px;

border: 1px solid #ddd;

margin-bottom: 12px;

}

img{

width: 120px;

height:120px;

}

#lupa{

width:20px;

height: 20px;

position:absolute;

left:15px;

top:75px;

}

/\* Sección de estilos para tabla de datos\*/

#tablaDatos {

border-collapse: collapse;

width: 100%;

border: 1px solid #ddd;

font-size: 18px;

}

/\* Sección de estilos para filas de campos y registros de tabla de datos\*/

#tablaDatos th, #tablaDatos td {

text-align: left;

padding: 12px;

}

#tablaDatos tr {

border-bottom: 1px solid #ddd;

}

/\* Sección de estilos con Pseudoclase:hover para cuando se seleccione fila de tabla se coloque de color gris\*/

#tablaDatos tr.header, #tablaDatos tr:hover {

background-color: #f1f1f1;

}

**En Javascript**

<script>

// Seccion donde creamos y definimos la programacion en jQuery para buscar el dato solicitado en la tabla de datos utilizando el evento keyup

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

$("#datos").on("keyup", function() **{**

var value = $(this).val().toLowerCase();

// Seccion donde definimos la programacion en jQuery para filtrar el dato solicitado en tabla de datos utilizando el método filter y el método indexOf()

$("#tablaDatos tr").filter(function() **{**

$(this).toggle($(this).text().toLowerCase().**indexOf**(value) > -1)

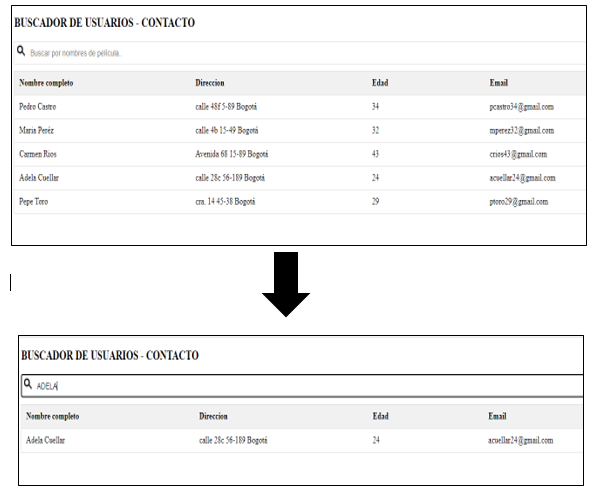
**});**

**});**

**});**

</script>

**RESULTADO FINAL**

****

Ahora debes de guardar este archivo de html con el nombre ***BUSCAR\_CONTACTO***.***html*** en la carpeta de html de la carpeta trabajos en clase, y enlazarlo en la pagina web en la sección menús de opciones utilizada en clase el semestre anterior como proyecto en clase.

**EDH (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.



# 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 programacion.

**Instrucciones**: Son un conjunto de lineas 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 librerias 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 pagina web HTML facilitando su programación en javascript y publicación web.

**Lenguaje** **hipertexto**: lengiaje 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.](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.

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/>

<https://appdelante.com/>