Desarrollo de Aplicaciones Web

Tema Nº5:JavaScript - II

Indicador de logro Nº5:Aplica el desarrollo del lado del cliente a través de JavaScript para el agrado de características interactivas a los sitios web

**TEMA 01 Teoría de los**

Imagen que contiene Icono

Descripción generada automáticamente

**TEMA Nº5:**

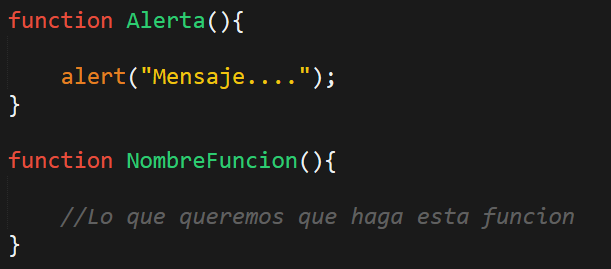
JavaScript - II

**Subtema 1.1:**

Funciones y Eventos

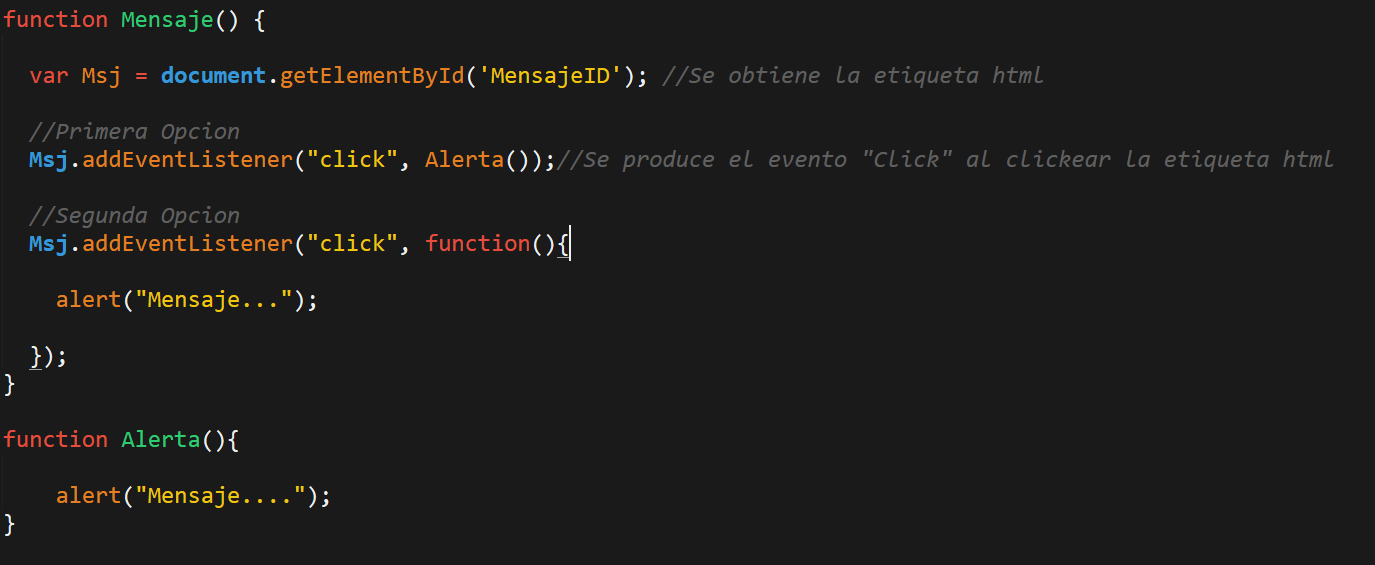
**MARCO TEÓRICO**

Las **funciones** en **JavaScript** son como los métodos en otros lenguajes. Nos permiten agrupar líneas de código, para que posteriormente, podamos hacer uso de ellas, ya sea llamándola en una etiqueta HTML o en otra función.



Los **eventos** son la manera que tenemos en **JavaScript** de controlar las acciones de los visitantes y definir un comportamiento cuando se produzcan. Con el manejo de eventos podemos hacer que nuestra página sea interactiva, porque con ello se puede responder a las interacciones con el usuario.

El **addEventListener** registra un evento a un objeto en específico. El objeto puede ser un elemento en un archivo, el mismo documento o una ventana.



Lista de eventos: <https://www.w3schools.com/jsref/dom_obj_event.asp>

La sintaxis es la siguiente: **LaEtiquetaQueSeUsara.AddEventListener(“ElEvento”, LaFuncion());**

**Subtema 1.5:**

Fundamentos de AJAX

AJAX son las siglas de Asynchronous JavaScript And Xml. No es un lenguaje de programación, sino un conjunto de tecnologías (HTML, JavaScript, CSS, DHTML, PHP, ASP.NET, JSP-XML) que nos permite hacer páginas web más interactivas.

**MATERIALES**

Para la experiencia a realizar se requiere lo siguiente:

1. EQUIPO

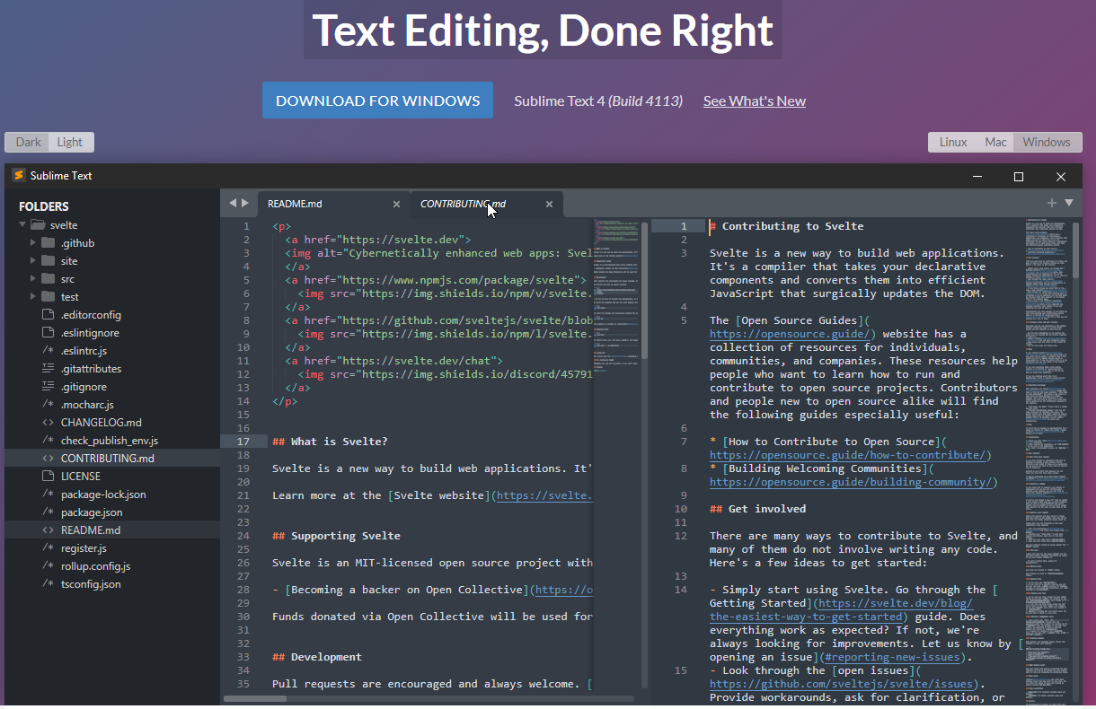
Se requiere tener un equipo de cómputo (PC o laptop).

1. HERRAMIENTAS:

* Sublime Text 3

**2. PROCEDIMIENTO**

Se creará un nuevo archivo, en el **Sublime Text**, en donde se aplicará lo aprendido en esta guía.

Herramienta de trabajo:

Se creará dos archivos, un archivo HTML y un archivo JAVASCRIPT

Para crear los archivos se puede ir a la pestaña File >> New File o podemos usar las teclas CRTL + N. Para guardar los archivos hacemos uso de las teclas CRTL + S.

**Ejemplo 1:** Se creará una clase Celular en la que se agregar nuevos Celulares manualmente y se mostraran mediante un botón.

Creamos las etiquetas a usar

<body>

<button type="button" onclick="document.write(Celular1.InfoCelular())" >

Mostrar Info del celular 1

</button>

<button type="button" onclick="document.write(Celular2.InfoCelular())">

Mostrar Info del celular 2

</button>

<button type="button" onclick="document.write(Celular3.InfoCelular())">

Mostrar Info del celular 3

</button><br><br>

<button type="button" onclick="Celular1.PresionarBotonEncendido()">

Presionar Boton Encendido del Celular 1

</button>

<button type="button" onclick="Celular2.PresionarBotonEncendido()">

Presionar Boton Encendido del Celular 2

</button>

<button type="button" onclick="Celular3.PresionarBotonEncendido()">

Presionar Boton Encendido del Celular 3

</button>

<br><br>

<button type="button" onclick="Celular1.TomarFoto()">

Tomar foto con el Celular 1</button>

<button type="button" onclick="Celular2.TomarFoto()">

Tomar foto con el Celular 2</button>

<button type="button" onclick="Celular3.TomarFoto()">

Tomar foto con el Celular 3</button>

</body>

Después se creará la clase Celular definiendo su constructor y sus métodos. El backtick (``, combinación del ALT + 96) o acento grave, sirve para poder llamar a expresiones como el: ${this.resolucion}, cosa que no podemos hacer si usamos la comilla doble (“ ”) o comida simple (‘ ’, también llamado apostrofe).

Class Celular{

constructor( color, peso, ram, marca, resolución )

{

this.color = color;

this.peso = peso;

this.ram = ram;

this.marca = marca;

this.resolucion = resolucion;

this.encendido = false;

}

PresionarBotonEncendido()

{

if(this.encendido == false){

alert("Celular encendido");

this.encendido = true

}

else{

alert("Celular apagado");

this.encendido = false;

}

}

TomarFoto()

{

if(this.encendido == true){

alert(`Foto tomada con una resolución de: ${this.resolucion}`);

}

else{

alert("No se puede tomar una foto porque el celular esta apagado");

}

}

InfoCelular()

{

Return `

Color: <b> ${this.color} </b><br>

Peso: <b> ${this.peso} </b><br>

Ram: <b> ${this.ram} </b><br>

Marca: <b> ${this.marca} </b><br>

Resolucion: <b> ${this.resolucion} </b><br>

`;

}

}

Con esto estaría listo, pero si deseamos que se muestre no usando los botones, podemos colocar lo siguiente

//Con esto creamos un nuevo Celular

Celular1= new Celular("Negro", "50g", "16gb", "Samsung", "1920x1080");

Celular2= new Celular("Blanco", "80g", "30gb", "iPhone", "4K");

Celular3= new Celular("Rosado", "65g", "8gb", "Motorola", "Full HD");

//Llamamos a las funciones

Celular1.PresionarBotonEncendido();

Celular1.TomarFoto();

Celular2.TomarFoto();

//Al nosotros usar el document.write para mostrar la info es necesario usar el backtick para poder

//Llamar al método y mostrar la información en la pagina

document.write(`

${Celular1.InfoCelular()}<br><br>

${Celular2.InfoCelular()}<br><br>

${Celular3.InfoCelular()}<br><br>

`)

**Ejemplo 2:** Haremos un formulario validando datos, comenzaremos con el HTML

//El evento onsubmit sirve para ejecutar la función al darle al botón guardar

//Si la función retorna false, no se ejecuta el submit. Sin embargo, si retorna true, se enviarán los datos

<form onsubmit="return ValidarFormulario();">

<label>Nombre</label><br>

<input type="text" Id="NombreID" ><br><br>

<label>Apellido</label><br>

<input type="text" Id="ApellidoID" ><br><br>

<label>Telefono</label><br>

<input type="text" Id="TelefonoID"><br><br>

<label>Email</label><br>

<input type="text" Id="EmailID"><br><br>

<button type="submit" >Guardar</button>

</form>

Ahora ejecutaremos el siguiente script para poder validar los campos

Para este script se usó expresiones regulares. Se uso este **/\w+@\w+\.+[a-z]/** para validar que se ingrese un formato de correo correcto, esta expresión se compone en:

**/contenido…/** 🡪 Se usa las dos barras azules al comienzo para señalar que es una expresión regular

**\w** 🡪 Esta clase de carácter sirve para señalar que se va a ingresar cualquier carácter alfanumérico, incluido el subrayado. Es equivalente a [A-Za-z0-9\_] lo que indica que se usara solamente el abecedario en mayúscula y minúscula, números del 0 al 9 y el guion bajo o subrayado

**+** 🡪 Este cuantificador sirve para señalar que va a haber más de un elemento que tenga detrás, por ejemplo /a+/ equivale a “aaaaaaaa”

**@** 🡪 Esto solo indica que se debe colocar una arroba

**\.** 🡪 Se coloca el \ antes del punto para indicar que es un carácter simple, no un carácter especial

**[a-z]** 🡪 Indica que solo está permitido usar letras desde la “a” hasta la “z” en minúscula.

Mas información sobre expresiones regulares, aquí: <https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular_Expressions>

function ValidarFormulario() {

var Nombre = document.getElementById('NombreID').value;

var Apellido = document.getElementById('ApellidoID').value;

var Email = document.getElementById('TelefonoID').value;

var Telefono = document.getElementById('EmailID').value;

FormatoCorreo = /\w+@\w+\.+[a-z]/;

if( Nombre.trim().length == 0 || Apellido.trim().length == 0 ||

Email.trim().length == 0 || Telefono.trim().length == 0){

alert("Todos los campos son obligatorios y no deben ser vacíos");

return false;

}

else if(!FormatoCorreo.test(Email)){

//Si no se cumple con el formato retornara false

//.test() valida si lo que esta dentro de los paréntesis tiene una

//coincidencia con la Expresión regular, en nuestro caso, FormatoCorreo

alert("El correo no tiene un formato valido");

return false;

}

else if(Telefono.trim().length > 9){

alert("No puedes agregar más de 9 dígitos en el teléfono");

return false;

}

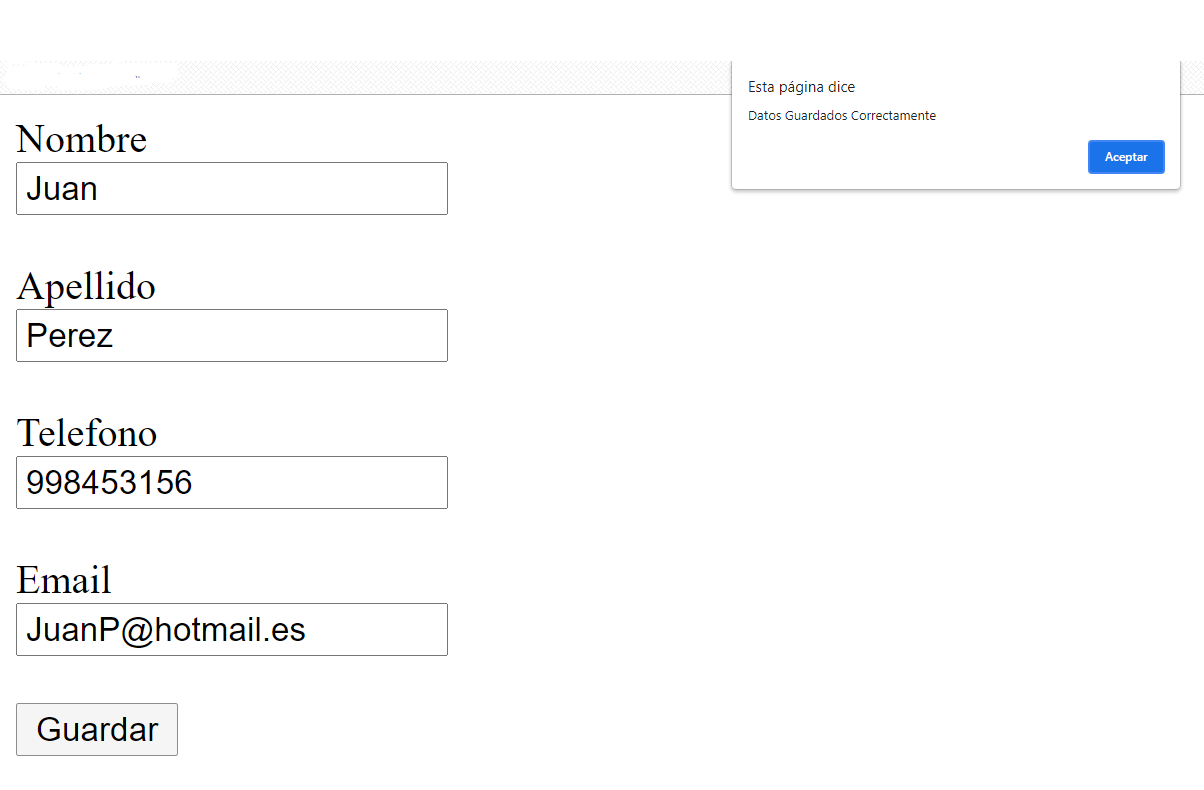
else{

alert("Datos guardados correctamente");//Por defecto retorna true

}

}

El resultado es:



**ACTIVIDAD VIRTUAL:**

1. **CUESTIONARIO TÉCNICO**

Revisa y analiza el tema desarrollado en la presente sesión, luego responde las siguientes preguntas propuestas:

* ¿Qué es AJAX?
* ¿Ventajas y desventajas de AJAX?
* ¿Porque es importante el POO en JavaScript?
* ¿Qué son las expresiones regulares y cuáles son sus caracteres especiales?
* Realice una App similar aplicando todo lo aprendido de esta guía.

1. **ENLACES DE REFERENCIA**

* https://desarrolloweb.com/articulos/1235.php
* https://lenguajejs.com/javascript/introduccion/funciones-basicas/
* https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular\_Expressions/Character\_Classes
* https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Regular\_Expressions/Groups\_and\_Ranges
* https://www.tutorialesprogramacionya.com/ajaxya/temarios/descripcion.php?punto=1&cod=8&inicio=0

1. **CONCLUSIONES DE LA EXPERIENCIA**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_