Especialidad: Técnico en Programación

Módulo 3. Submódulo2. Desarrolla aplicaciones que se ejecutan en el cliente.

Prof: Hilda Lucía Rodríguez Gómez

Competencia Profesional: Usa JavaScript para manejar eventos

Actividades sobre Formularios:

1.- Copia en bloc de notas o en brackets las etiquetas básicas para formularios, del listado 1. Agrega cada elemento <input> especificado en la primer tabla de este documento (email, search, url, tel,number, range… etc) y pruébalos en el navegador. Observación, en los tipos "number" y "range" se pueden especificar los atributos min, max y step con el siguiente formato:

```
<input type="range" name="numero" id="numero" min="0" max="10" step="5">
```

Listado 1:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section>
<form name="miformulario" id="miformulario" method="get">
<input type="text" name="nombre" id="nombre">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Agregar etiqueta <input> para cada tipo proporcionado

Cambiar nombre e id, conforme se agreguen los diferentes type

2.- En el mismo html que tienes, prueba los siguientes atributos (placeholder, required, multiple, autofocus, pattern), agregándolos en su etiqueta correspondiente:

```
<input type="search" name="busqueda" id="busqueda" placeholder="escriba su búsqueda">

<input type="email" name="miemail" id="miemail" required>

<input type="email" name="miemail" id="miemail" multiple>

<input type="search" name="busqueda" id="busqueda" autofocus>

<input pattern=" [0-9]{5}" name="codigopostal" id="codigopostal" title="inserte los 5 números de su código postal">
```

3. Atributo form: nos permite declarar elementos para un formulario fuera del ámbito de las etiquetas **<form>**. Hasta ahora, para construir un formulario teníamos que escribir las etiquetas **<form>** de apertura y cierre y luego declarar cada elemento del formulario entre ellas. En HTML5 podemos insertar los elementos en cualquier parte del código y luego hacer referencia al formulario que pertenecen usando su nombre y el atributo **form.**

Crea un segundo archivo de html y prueba el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<nav>
<input type="search" name="busqueda" id="busqueda" form="formulario">
</nav>
<section>
<form name="formulario" id="formulario" method="get">
<input type="text" name="nombre" id="nombre">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

4. Probar los nuevos elementos para formularios. Regresa al primer código html, agrega los siguientes bloques de etiquetas y pruébalos en el navegador:

```
<datalist id="informacion">
<option value=”123123123” label=”Teléfono 1”>
<option value=”456456456” label=”Teléfono 2”>
</datalist>
<input type="tel" name="telefono" id="telefono" list="informacion">
```

Actividades para Validación :

Código 1

Especificaciones del código que se muestra a continuación: Se establecen dos campos para el nombre y apellido del usuario. Sin embargo, el formulario mostrará error, cuando ambos campos se encuentran vacíos.

-En casos como éste no es posible usar el atributo required debido a que no sabemos cuál campo decidirá utilizar el usuario. En este caso podemos utilizar el método ***setCustomValidity()*** *-*

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
<script>
function iniciar(){
nombre1=document.getElementById("nombre");
nombre2=document.getElementById("apellido");
nombre1.addEventListener("input", validacion, false);
nombre2.addEventListener("input", validacion, false);
validacion();
}
function validacion(){
if(nombre1.value=='' && nombre2.value==''){
nombre1.setCustomValidity('inserte al menos un nombre');
nombre1.style.background='#FFDDDD';
}else{
nombre1.setCustomValidity('');
nombre1.style.background='#FFFFFF';
}
}
window.addEventListener("load", iniciar, false);
</script>
</head>
<body>
```

```
<section>
<form name="registracion" method="get">
Nombre:
<input type="text" name="nombre" id="nombre">
Apellido:
<input type="text" name="apellido" id="apellido">
<input type="submit" id="send" value="ingresar">
</form>
</section>
</body>
</html>
```

Código 2

Uso del evento invalid. Creamos un nuevo formulario con tres campos para ingresar el nombre de usuario, un email y un rango de 20 años de edad.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
<script>
function iniciar(){
edad=document.getElementById("miedad");
edad.addEventListener("change", cambiarrango, false);
document.informacion.addEventListener("invalid", validacion, true);
document.getElementById("enviar").addEventListener("click", enviar, false);
}
function cambiarrango(){
var salida=document.getElementById("rango");
var calc=edad.value-20;
if(calc<20){
calc=0;
edad.value=20;
}
salida.innerHTML=calc+' a '+edad.value;
}
function validacion(e){
var elemento=e.target;
elemento.style.background='#FFDDDD';
}
function enviar(){
var valido=document.informacion.checkValidity();
if(valido){
document.informacion.submit();
}
}
window.addEventListener("load", iniciar, false);
</script>
</head>
<body>
<section>
<form name="informacion" method="get">
Usuario:
<input pattern="[A-Za-z]{3,}" name="usuario" id="usuario"
maxlength="10" required>
Email:
<input type="email" name="miemail" id="miemail" required>
Rango de Edad:
<input type="range" name="miedad" id="miedad" min="0" max="80" step="20" value="20">
<output id="rango">0 a 20</output>
<input type="button" id="enviar" value="ingresar">
</form>
</section>
</body>
</html>
```

Código 3

Validación en tiempo real. Cuando abrimos el archivo con la plantilla del Listado 6-24 en el navegador, podremos notar que no existe una validación en tiempo real. Los campos son sólo validados cuando el botón "ingresar" es presionado. Para hacer más práctico nuestro sistema personalizado de validación, tenemos que aprovechar los atributos provistos por el objeto **ValidityState**.

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
<script>
function iniciar(){
edad=document.getElementById("miedad");
edad.addEventListener("change", cambiarrango, false);
document.informacion.addEventListener("invalid", validacion, true);
document.getElementById("enviar").addEventListener("click", enviar, false);
document.informacion.addEventListener("input", controlar, false);
}
function cambiarrango(){
var salida=document.getElementById("rango");
var calc=edad.value-20;
if(calc<20){
calc=0;
edad.value=20;
}
salida.innerHTML=calc+' a '+edad.value;
}
function validacion(e){
var elemento=e.target;
elemento.style.background='#FFDDDD';
}
function enviar(){
var valido=document.informacion.checkValidity();
if(valido){
document.informacion.submit();
}
}
function controlar(e){
var elemento=e.target;
if(elemento.validity.valid){
elemento.style.background='#FFFFFF';
}else{
elemento.style.background='#FFDDDD';
}
}
window.addEventListener("load", iniciar, false);
</script>
</head>
<body>
<section>
<form name="informacion" method="get">
Usuario:
<input pattern="[A-Za-z]{3,}" name="usuario" id="usuario" maxlength="10" required>
Email:
<input type="email" name="miemail" id="miemail" required>
Rango de Edad:
<input type="range" name="miedad" id="miedad" min="0" max="80" step="20" value="20">
<output id="rango">0 a 20</output>
<input type="button" id="enviar" value="ingresar">
</form>
</section>
</body>
</html>
```

Por último, vamos a usar estados de validación para mostrar un mensaje de error personalizado. Reemplaza la función **enviar()** del código anterior con la nueva función enviar() presentada a continuación y abre el archivo en el navegador.

```
function enviar(){
var elemento=document.getElementById("usuario");
var valido=document.informacion.checkValidity();
if(valido){
document.informacion.submit();
}else if(elemento.validity.patternMismatch ||
elemento.validity.valueMissing){
alert('el nombre de usuario debe tener mínimo de 3 caracteres');
}
}
```

Para controlar estos estados de validación, debemos utilizar la sintaxis **elemento.validity.estado** (donde **estado** es cualquiera de los valores listados arriba). Podemos aprovechar estos atributos para saber exactamente que originó el error en un formulario.