INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

INGENIERÍA EN SISTEMAS COMPUTACIONALES

MATERIA: TECNOLOGÍAS PARA LA WEB

PROFESORA: RIVERA DE LA ROSA MONICA

PRESENTA:

RAMIREZ BENITEZ BRAYAN

GRUPO: 2CM14

ACCESO A ELEMENTOS DEL FORMULARIO

CIUDAD DE MEXICO A 25 DE OCTUBRE DE 2021

Obtener el valor de los campos de formulario

La mayoría de las técnicas JavaScript relacionadas con los formularios requieren leer y/o modificar el valor de los campos del formulario.

Cuadro de texto y textarea

El valor del texto mostrado por estos elementos se obtiene y se establece directamente mediante la propiedad value.

```
<input type="text" id="texto" />
var valor = document.getElementById("texto").value;


<textarea id="parrafo"></textarea>
var valor = document.getElementById("parrafo").value;
```

Radiobutton

Cuando se dispone de un grupo de radiobuttons, generalmente no se quiere obtener el valor del atributo value de alguno de ellos, sino que lo importante es conocer cuál de todos los radiobuttons se ha seleccionado. La propiedad checked devuelve true para el radiobutton seleccionado y false en cualquier otro caso. Si por ejemplo se dispone del siguiente grupo de radiobuttons:

```
<input type="radio" value="si" name="pregunta" id="pregunta_si"/> SI
<input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO
<input type="radio" value="nsnc" name="pregunta" id="pregunta_nsnc"/> NS/NC
```

El siguiente código permite determinar si cada radiobutton ha sido seleccionado o no:

```
var elementos = document.getElementsByName("pregunta");

for(var i=0; i<elementos.length; i++) {
  alert(" Elemento: " + elementos[i].value + "\n Seleccionado: " + elementos[i].checked);
}
```

Checkbox

Los elementos de tipo checkbox son muy similares a los radiobutton, salvo que en este caso se debe comprobar cada checkbox de forma independiente del resto. El motivo es que los grupos de radiobutton son mutuamente excluyentes y sólo se puede seleccionar uno de ellos cada vez. Por su parte, los checkbox se pueden seleccionar de forma independiente respecto de los demás.

Si se dispone de los siguientes checkbox:

```
<input type="checkbox" value="condiciones" name="condiciones" id="condiciones"/> He leído y
acepto las condiciones
<input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/> He leído la p
olítica de privacidad
```

Utilizando la propiedad checked, es posible comprobar si cada checkbox ha sido seleccionado:

```
var elemento = document.getElementById("condiciones");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);

elemento = document.getElementById("privacidad");
alert(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.checked);
```

Select

Las listas desplegables (<select>) son los elementos en los que es más difícil obtener su valor. Si se dispone de una lista desplegable como la siguiente:

```
<select id="opciones" name="opciones">
  <option value="1">Primer valor</option>
  <option value="2">Segundo valor</option>
  <option value="3">Tercer valor</option>
  <option value="4">Cuarto valor</option>
</select>
```

En general, lo que se requiere es obtener el valor del atributo value de la opción (<option>) seleccionada por el usuario. Obtener este valor no es sencillo, ya que se deben realizar una serie de pasos. Además, para obtener el valor seleccionado, deben utilizarse las siguientes propiedades:

- options, es un array creado automáticamente por el navegador para cada lista desplegable y que contiene la referencia a todas las opciones de esa lista. De esta forma, la primera opción de una lista se puede obtener mediante document.getElementById("id_de_la_lista").options[0].
- selectedIndex, cuando el usuario selecciona una opción, el navegador actualiza automáticamente el valor de esta propiedad, que guarda el índice de la opción seleccionada. El índice hace referencia al array options creado automáticamente por el navegador para cada lista.

```
// Obtener la referencia a la lista
var lista = document.getElementById("opciones");

// Obtener el índice de la opción que se ha seleccionado
var indiceSeleccionado = lista.selectedIndex;
// Con el índice y el array "options", obtener la opción seleccionada
var opcionSeleccionada = lista.options[indiceSeleccionado];

// Obtener el valor y el texto de la opción seleccionada
var textoSeleccionado = opcionSeleccionada.text;
var valorSeleccionado = opcionSeleccionada.value;

alert("Opción seleccionada: " + textoSeleccionado + "\n Valor de la opción: " + valorSelecc
ionado);
```

Como se ha visto, para obtener el valor del atributo value correspondiente a la opción seleccionada por el usuario, es necesario realizar varios pasos. No obstante, normalmente se abrevian todos los pasos necesarios en una única instrucción:

```
var lista = document.getElementById("opciones");

// Obtener el valor de la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].value;

// Obtener el texto que muestra la opción seleccionada
var valorSeleccionado = lista.options[lista.selectedIndex].text;
```

Lo más importante es no confundir el valor de la propiedad selectedIndex con el valor correspondiente a la propiedad value de la opción seleccionada. En el ejemplo anterior, la primera opción tiene un value igual a 1. Sin embargo, si se selecciona esta opción, el valor de selectedIndex será 0, ya que es la primera opción del array options (y los arrays empiezan a contar los elementos en el número 0).

Evitar el envío duplicado de un formulario

Uno de los problemas habituales con el uso de formularios web es la posibilidad de que el usuario pulse dos veces seguidas sobre el botón "Enviar". Si la conexión del usuario es demasiado lenta o la respuesta del servidor se hace esperar, el formulario original sigue mostrándose en el navegador y por ese motivo, el usuario tiene la tentación de volver a pinchar sobre el botón de "Enviar".

En la mayoría de los casos, el problema no es grave e incluso es posible controlarlo en el servidor, pero puede complicarse en formularios de aplicaciones importantes como las que implican transacciones económicas.

Por este motivo, una buena práctica en el diseño de aplicaciones web suele ser la de deshabilitar el botón de envío después de la primera pulsación. El siguiente ejemplo muestra el código necesario:

```
<form id="formulario" action="#">
  ...
  <input type="button" value="Enviar" onclick="this.disabled=true; this.value='Enviand
o...'; this.form.submit()" />
</form>
```

Cuando se pulsa sobre el botón de envío del formulario, se produce el evento onclick sobre el botón y, por tanto, se ejecutan las instrucciones JavaScript contenidas en el atributo onclick:

1. En primer lugar, se deshabilita el botón mediante la instrucción this.disabled = true;. Esta es la única instrucción necesaria si sólo se quiere deshabilitar un botón.
2. A continuación, se cambia el mensaje que muestra el botón. Del original "Enviar" se pasa al más adecuado "Enviando..."
3. Por último, se envía el formulario mediante la función submit() en la siguiente instrucción: this.form.submit()

El botón del ejemplo anterior está definido mediante un botón de tipo <input type="button" />, ya que el código JavaScript mostrado no funciona correctamente con un botón de tipo <input type="submit" />. Si se utiliza un botón de tipo submit, el botón se deshabilita antes de enviar el formulario y por tanto el formulario acaba sin enviarse.

Prompt

Un cuadro de diálogo utilizando prompt() contiene un cuadro de texto de una línea, un botón «Cancelar» y un botón «Aceptar».

Al pulsar el botón «Aceptar» retorna el texto que el usuario ha introducido en el cuadro de texto.

El cuadro de diálogo que ofrece prompt() es una ventana modal que previene que el usuario acceda al resto de la página hasta que el cuadro de diálogo sea cerrado. Por este motivo, no se recomienda abusar de este tipo de ventanas ya que pueden causar molestias a los usuarios.

Esta función puede recibir 2 parámetros:

- Parámetro 1 = Mensaje: es una cadena de texto que se mostrará al usuario. Este parámetro es opcional y puede ser omitido si no se necesita mostrar nada en la ventana.
- Parámetro 2 = defecto: es una cadena de texto que contiene el valor predeterminado para el texto de entrada. Es opcional.

Nótese que el resultado es una cadena de texto. Esto significa que a veces se deberá hacer una conversión al valor introducido por el usuario. Por ejemplo, si la respuesta debe ser un valor numérico, se debe hacer la conversión del valor a tipo Number. var aNumber = Number(window.prompt("Type a number", ""));

```
let name = prompt('Cual es tu nombre?');
let age = prompt('Cual es tu edad?');

const currentDate = new Date();
const year = currentDate.getFullYear() - Number(age);

document.getElementById('content').innerHTML = `<h2>Bienvenido ${name} de ${age} años.<h2/><h3>Naciste el año ${year}.</h3>`;
```

En tan solo unas pocas líneas de código tenemos una interfaz que aparece antes de cargar la página y que permite realizar preguntas al usuario antes de empezar una aplicación o antes de mostrarle los contenidos. De esta forma tan simple, en el ejemplo en funcionamiento, realiza 2 preguntas al usuario antes de entrar en la página web. Para saber cuál es su nombre y que edad tiene. Hace un pequeño cálculo para averiguar su año de nacimiento y, después, muestra las variables en el contenedor principal de la página. Dentro del cuerpo de la página o dentro de la etiqueta <body>, se ha agregado el contenedor vacío que recibe los datos mediante la manipulación del DOM con JavaScript.

```
<div id="content" class="col-lg-12">

</div>
```

Document.write

Uno de los comandos básicos de JavaScript es document.write. Esto imprime el texto especificado en la página.

Nota: dado que document.write escribe directo al hilo (stream) de un documento, la llamada a document.write en un documento ya cargado automáticamente ejecuta document.open, lo cual limpiará todo el contenido del documento en cuestión.

Sintaxis

```
document.write(texto);
```

Donde texto es una cadena de texto que contiene el texto a ser impreso en el documento.

Ejemplo

```
<html>

<head>
  <title>Ejemplo de write</title>

  <script>
    function nuevoContenido() {
      alert("carga el contenido nuevo");
      document.open();
      document.write("<h1>Quita el contenido viejo - Agrega el contenido nuevo!</h1>");
      document.close();
    }
  </script>
</head>

<body onload="nuevoContenido();">
  <p>Algo de contenido original del documento</p>
</body>

</html>
```

Escribir a un documento que ya tiene contenido cargado previamente sin llamar a document.open(), automáticamente hará una llamada a document.open(). Después de haber finalizado la escritura del documento, es recomendable llamar a document.close(), para informar al navegador que la carga de la página ya ha terminado. El texto que escribas allí es convertido a la estructura tipificada de HTML dentro del modelo estructural del

documento. En el ejemplo de más arriba, el elemento h1 se convierte en un nodo dentro del documento.

Si la llamada a document.write() se ejecuta dentro de una etiqueta <script> incluído en el HTML, entonces la llamada a document.open() nunca ocurrirá. Por ejemplo:

```
<script>
  document.write("<h1>Título Principal</h1>")
</script>
```