Tarea

Función prompt, doment.write

Acceso de elementos de un formulario

- Mora Ayala Jose Antonio

CONTENIDO

Tarea 1

Acceso a valores en formularios 3

1.1 El objeto *form* 3

1.2 Propiedades principales del objeto *form* 3

1.2.1 Sintaxis básica 4

1.3.1 Sintaxis básica 4

16.2 UTILIDADES BÁSICAS PARA FORMULARIOS 13

16.2.1 OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO 13

16.2.1.1 CUADRO DE TEXTO Y TEXTAREA 13

16.2.1.2 RADIOBUTTON 13

16.2.1.3 CHECKBOX 14

16.2.1.4 SELECT 15

16.2.2 ESTABLECER EL FOCO EN UN ELEMENTO 16

16.2.3 EVITAR EL ENVÍO DUPLICADO DE UN FORMULARIO 18

16.2.4 LIMITAR EL TAMAÑO DE CARACTERES DE UN TEXTAREA 19

16.2.5 RESTRINGIR LOS CARACTERES PERMITIDOS EN UN CUADRO DE TEXTO 20

16.3 VALIDACIÓN 22

16.3.1 VALIDAR UN CAMPO DE TEXTO OBLIGATORIO 24

16.3.2 VALIDAR UN CAMPO DE TEXTO CON VALORES NUMÉRICOS 25

16.3.3 VALIDAR QUE SE HA SELECCIONADO UNA OPCIÓN DE UNA LISTA 26

16.3.4 VALIDAR UNA DIRECCIÓN DE EMAIL 26

16.3.5 VALIDAR UNA FECHA 27

16.3.6 VALIDAR UN NÚMERO DE DNI 27

16.3.9 VALIDAR QUE UN RADIOBUTTON HA SIDO SELECCIONADO 29

Función document.write 30

prompt 31

ACCESO A VALORES EN FORMULARIOS

1.1 El objeto *form*

El objeto form es un sub-objeto del objeto *document* y este a su vez, lo es del objeto *window*.

Así como para crear una página en HTML se utilizan las etiquetas <HTML> Y </HTML>, lo mismo sucede con un formulario: el formulario debe estar contenido entre las etiquetas **<form>** y **</form>**

En principio la sintaxis básica para referirnos a un formulario sería:

window.document.forms.nombre_del_formulario

En la que tranquilamente pueden prescindirse de ***window*** y ***forms*** ya que el navegador toma al formulario como un objeto en sí mismo. De la misma forma, tambien puede prescindirse de ***document***. Pero esta omisión solo se hará si queremos referirnos a un formulario en particular (por ejemplo a un formulario llamado "datos"). Pero al momento de referirnos a "todos los formularios de una página", solo se podrá prescindir del objeto window.

De todas formas, iremos viendo la aplicación de este tipo de sintaxis con los próximos ejemplos y a medida que avancemos.

1.2 Propiedades principales del objeto *form*

El objeto *form* posee las siguientes propiedades:

propiedad	**descripción**
name	es el nombre único del formulario.
action	es el lugar al cual se envía el formulario para ser procesado. El *action* define la URL a la cual se envía dicho formulario.
method	método de envío de los datos insertados en un formulario. El *method* puede ser: **GET** = envía los datos en una cadena "visible". Conveniente para enviar pocos datos. **POST** = envía los datos en forma "invisible". Conveniente para enviar una gran cantidad de datos.

target	define la ventana o marco (frame) en la que se mostrarán o procesarán los resultados del formulario. El valor es el mismo que el utilizado en HTML (blank, self, top, nombre_marco, etc..)

1.2.1 Sintaxis básica

<form name="nombre_formulario" action="procesar.asp" method="POST" target="_blank">

.....campos....

</form>

1.3 Métodos del objeto *form*

El objeto form posee dos métodos:

método	descripción
submit	envía el formulario.
reset	restablece el formulario a los valores por defecto.

1.3.1 Sintaxis básica

<form name="nombre_formulario" action="procesar.asp" method="POST" target="_blank">

.....campos....

<input type="submit" value="enviar formulario">

<input type="reset" value="borrar">

</form>

1.4 Ejemplo de aplicación

Con estos ejemplos veremos la utilización de la propiedad ***method*** y de los métodos ***submit*** y ***reset***.

Method POST

El código....

```
<form name="datos" action="ejemplos/procesar.asp" method="POST" target="_blank">
Escribe tu nombre: <input type="text" name="nombre"><br>
<input type="submit" value="enviar formulario"><br>
<input type="reset" value="borrar">
</form>
```

El resultado...

Escribe tu nombre:

enviar formulario borrar

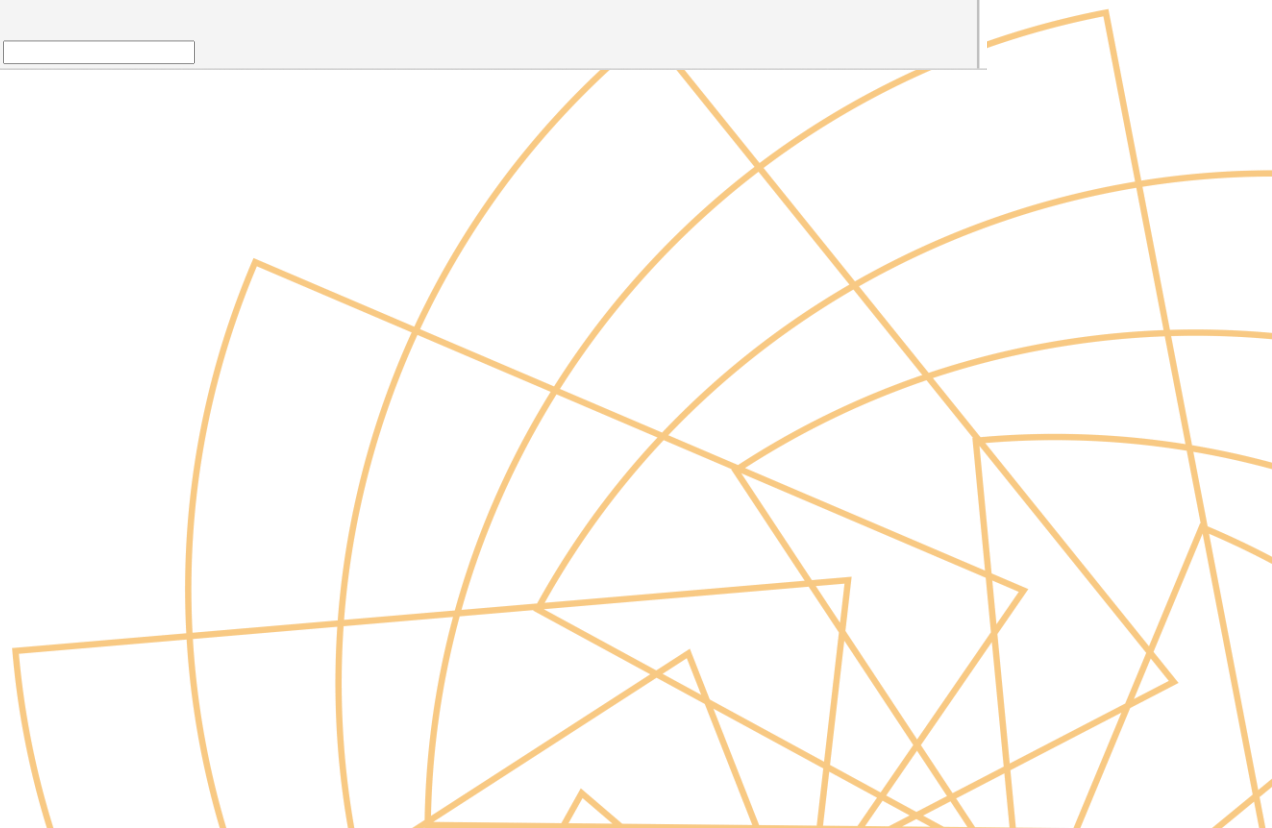
Method GET

El código....

```
<form name="datos1" action="ejemplos/procesar1.asp" method="GET" target="_blank">
Escribe tu nombre: <input type="text" name="nombre"><br>
<input type="submit" value="enviar formulario"><br>
<input type="reset" value="borrar">
</form>
```

El resultado...

Escribe tu nombre:



TODOS LOS FORMULARIOS EN FORMS

undefined - Untitled-2

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo aprenderaprogramar.com</title>
    <meta charset="utf-8" />
    <style type="text/css">
      label {
        display: block;
        margin: 5px;
      }
    </style>
    <script type="text/javascript">
      window.onload = function () {
        var ejemplo = document.getElementById("ejemplo");
        ejemplo.addEventListener("click", ejecutarEjemplo);
      };
      function ejecutarEjemplo() {
        var formularios = document.forms;
        alert(
          "El número de formularios en el documento es: " + formularios.length
        );
      }
    </script>
  </head>
  <body>
    <div id="cabecera">
      <h2>Cursos aprenderaprogramar.com</h2>
      <h3>Ejemplos JavaScript</h3>
    </div>
    <div style="color: blue; margin: 20px" id="ejemplo">Pulsa aquí</div>
    <form name="formularioContacto" method="get" action="accion1.html">
      <h2>Formulario de contacto</h2>
      <label
        >Nombre:<input id="nombreFormContacto" type="text" name="nombre"
      /></label>
      <label
        >Apellidos:<input
          id="apellidosFormContacto"
          type="text"
          name="apellidos"
      /></label>
      <label><input id="botonEnvio1" type="submit" value="Enviar" /></label
>
    </form>
    <form name="formularioReclamacion" method="get" action="accion2.html">
      <h2>Formulario de reclamación</h2>
      <label
        >Motivo reclamación:<input
          id="motivoFormReclama"
          type="text"
          name="motivo"
      /></label>
      <label
        >Fecha del hecho:<input id="fechaFormReclama" type="text" name="fec
ha"
      /></label>
      <label><input id="botonEnvio2" type="submit" value="Enviar" /></label
>
    </form>
  </body>
</html>

```

La propiedad forms del objeto document devuelve una colección de objetos tipo array conteniendo todos los formularios que existan en el documento HTML, con índice 0 para el primer formulario por orden de aparición, índice 1 para el siguiente y así sucesivamente.

En este ejemplo vemos cómo podemos acceder a document.forms y obtener el número de formularios existentes en el documento HTML:

TODOS LOS CAMPOS DE FORMULARIOS EN ELEMENTS

Si prestamos atención a los campos de un formulario, cada uno de estos campos puede tener hasta dos identificadores importantes. Por ejemplo:

```
<input id="nombreFormContacto" type="text" name="nombre" />
```

Estos identificadores son el id, que identifica de forma única a un elemento dentro de un documento html (es decir, el id es único en toda la página web), y el atributo name, que identifica de forma única a un elemento dentro de un formulario (es decir, no habrá dos elementos dentro de un formulario con el mismo atributo name, pero sí puede haber elementos en distintos formularios que tengan el mismo atributo name.

Es frecuente que id y name tengan un mismo valor, por ejemplo <input id="nombre" type="text" name="nombre" />, pero no siempre ocurre así.

El atributo id, si existe, nos permitiría acceder a este elemento del formulario usando el método getElementById('valorDeId').

La propiedad name posiblemente no nos sea útil para acceder con getElementsByName debido a que pueden existir varios elementos con el mismo name, sin embargo sí nos va a ser útil con métodos para trabajar con formularios de los que nos provee JavaScript de los que vamos a ir hablando.

La propiedad elements de cada objeto form obtenido mediante document.forms devuelve una colección de objetos tipo array conteniendo todos los campos de formulario que existan en el formulario concreto al que hagamos alusión, con índice 0 para el primer elemento por orden de aparición, índice 1 para el siguiente y así sucesivamente.

Cada campo de formulario es para JavaScript un "HTML Object" y como tal tiene propiedades a las que podemos acceder. Por ejemplo name es una propiedad, value otra propiedad, id otra propiedad, etc.

Analiza este código y comprueba sus resultados en tu navegador para ver cómo elements nos permite acceder a los campos de formulario.

undefined - Untitled-2

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo aprenderaprogramar.com</title>
    <meta charset="utf-8" />
    <style type="text/css">
      label {
        display: block;
        margin: 5px;
      }
    </style>
    <script type="text/javascript">
      window.onload = function () {
        var ejemplo = document.getElementById("ejemplo");
        ejemplo.addEventListener("click", ejecutarEjemplo);
      };
      function ejecutarEjemplo() {
        var formularios = document.forms;
        alert(
          "El número de formularios en el documento es: " + formularios.length
        );
      }
    </script>
  </head>
  <body>
    <div id="cabecera">
      <h2>Cursos aprenderaprogramar.com</h2>
      <h3>Ejemplos JavaScript</h3>
    </div>
    <div style="color: blue; margin: 20px" id="ejemplo">Pulsa aquí</div>
    <form name="formularioContacto" method="get" action="accion1.html">
      <h2>Formulario de contacto</h2>
      <label
        >Nombre:<input id="nombreFormContacto" type="text" name="nombre"
      /></label>
      <label
        >Apellidos:<input
          id="apellidosFormContacto"
          type="text"
          name="apellidos"
      /></label>
      <label><input id="botonEnvio1" type="submit" value="Enviar" /></label
    >
    </form>
    <form name="formularioReclamacion" method="get" action="accion2.html">
      <h2>Formulario de reclamación</h2>
      <label
        >Motivo reclamación:<input
          id="motivoFormReclama"
          type="text"
          name="motivo"
      /></label>
      <label
        >Fecha del hecho:<input id="fechaFormReclama" type="text" name="fecha"
      /></label>
      <label><input id="botonEnvio2" type="submit" value="Enviar" /></label
    >
    </form>
  </body>
</html>
```

El resultado esperado es que cuando pulsemos en el texto “Pulsa aquí” se muestre por pantalla lo siguiente:

Elemento 0 del formulario 1 tiene id: nombreFormContacto y name: nombre

Elemento 1 del formulario 1 tiene id: apellidosFormContacto y name: apellidos

Elemento 2 del formulario 1 tiene id: botonEnvio1 y name:

Elemento 0 del formulario 2 tiene id: motivoFormReclama y name: motivo

Elemento 1 del formulario 2 tiene id: fechaFormReclama y name: fecha

Elemento 2 del formulario 2 tiene id: botonEnvio2 y name:

Vemos cómo usando elements hemos accedido a cada uno de los elementos HTML dentro del formulario y mostrado sus propiedades. Vemos cómo los campos name correspondientes a los botones de envío quedan vacíos debido a que estos elementos input no tienen establecido atributo name.

ACCESO DIRECTO A FORMULARIOS MEDIANTE EL ATRIBUTO NAME

Aunque podamos acceder a cualquier elemento mediante el uso de document.forms[indice1].elements[indice2], el acceso mediante índices numéricos posiblemente no resulta cómodo. Además, si introducimos nuevos elementos en los formularios o nuevos formularios, los índices numéricos cambiarán y esto afectaría al diseño de nuestro código.

JavaScript provee una forma cómoda de acceder a los formularios a la que denominamos "acceso directo al formulario a través de su atributo name". Para ello nos basamos en que cada formulario se mantiene como una propiedad de document a la que se puede acceder escribiendo simplemente:

```
document.valorAtributoNameDelFormulario
```

De esta forma podemos obtener una referencia a un formulario simplemente con una expresión del tipo: var formulario1 = document.formularioContacto; donde formularioContacto es el atributo name de un formulario dentro del documento HTML.

ACCESO DIRECTO A ELEMENTOS DE FORMULARIOS MEDIANTE EL ATRIBUTO NAME

Del mismo modo que podemos acceder a un formulario usando el valor de su atributo name, podemos acceder a elementos de un formulario usando el atributo name de estos elementos con una sintaxis de este tipo:

```
document.valorAtributoNameDelFormulario.valorAtributoNameDelElementoHTML
```

De esta forma podemos obtener una referencia a un elemento simplemente con una expresión del tipo: var elemento1 = document.formularioContacto.apellidos; donde formularioContacto es el atributo name de un formulario dentro del documento HTML y apellidos es el atributo name de un elemento input dentro del formulario. Aunque en una primera lectura puede parecer un trabalenguas, leyéndolo un par de veces y viendo un ejemplo se comprenderá con facilidad.

Veamos el ejemplo. Escribe este código, guárdalo con un nombre como ejemplo.html y visualiza los resultados que se obtienen.

En este ejemplo var formulario2 = document.formularioReclamacion; nos devuelve el objeto HTML que es el form cuyo atributo name es formularioReclamacion. A su vez document.formularioReclamacion.apellidos nos devuelve el elemento HTML que es el input cuyo atributo name es apellidos.

Estos mismos resultados podrían conseguirse combinando los distintos métodos de acceso a nodos del DOM que ofrece JavaScript (document.getElementById, getElementsByName, childnodes, etc.) o los arrays document.forms y document.elements, pero el acceso basado en nombres resultará cómodo si hemos creado una buena definición de nombres en nuestro documento HTML.

undefined - Untitled-2

```
<!DOCTYPE html>
<html>
  <head>
    <title>Ejemplo aprenderaprogramar.com</title>
    <meta charset="utf-8" />
    <style type="text/css">
      label {
        display: block;
        margin: 5px;
      }
    </style>
    <script type="text/javascript">
      window.onload = function () {
        var ejemplo = document.getElementById("ejemplo");
        ejemplo.addEventListener("click", ejecutarEjemplo);
      };
      function ejecutarEjemplo() {
        var formulario1 = document.formularioContacto;
        dimeDatos(formulario1);
        var formulario2 = document.formularioReclamacion;
        dimeDatos(formulario2);
      }
      function dimeDatos(formulario) {
        var msg = "";
        msg =
          msg +
          "\n\nElemento input del formulario " +
          formulario.name +
          " tiene id: " +
          formulario.nombre.id;
        msg =
          msg +
          "\n\nOtro elemento input del formulario " +
          formulario.name +
          " tiene id: " +
          formulario.apellidos.id;
        alert(msg + "\n\n");
      }
    </script>
  </head>
  <body>
    <div id="cabecera">
      <h2>Cursos aprenderaprogramar.com</h2>
      <h3>Ejemplos JavaScript</h3>
    </div>
    <div style="color: blue; margin: 20px" id="ejemplo">Pulsa aquí</div>
    <form name="formularioContacto" method="get" action="accion1.html">
      <h2>Formulario de contacto</h2>
      <label
        >Nombre:<input id="nombreFormContacto" type="text" name="nombre"
      /></label>
      <label
        >Apellidos:<input
          id="apellidosFormContacto"
          type="text"
          name="apellidos"
      /></label>
      <label><input id="botonEnvio1" type="submit" value="Enviar" /></label
>
    </form>
    <form name="formularioReclamacion" method="get" action="accion2.html">
      <h2>Formulario de reclamación</h2>
      <label
        >Nombre:<input id="nombreFormReclamacion" type="text" name="nombre"
      /></label>
      <label
        >Apellidos:<input
          id="apellidosFormReclamacion"
          type="text"
          name="apellidos"
      /></label>
      <label><input id="botonEnvio2" type="submit" value="Enviar" /></label
>
    </form>
  </body>
</html>

```

ACCESO CON SINTAXIS MIXTA FORMS – ELEMENTS – NAME – ID

Hay más forma de acceder a los elementos de un formulario, e incluso al formulario en sí.

Podemos acceder a formularios o a los elementos de un formulario con una sintaxis como:

```
var formulario1 = document.forms['valorAtributoIdDelFormulario'];

var elemento1 = formulario1.elements['valorAtributoIdDelElemento']
```

Por ejemplo var formulario1 = document.forms['form1']; nos permite acceder a un formulario cuyo atributo id es form1.

Igualmente podemos acceder a formularios o a los elementos de un formulario con una sintaxis como:

```
var formulario1 = document.forms['valorAtributoNameDelFormulario'];

var elemento1 = formulario1.elements['valorAtributoNameDelElemento'];
```

Por ejemplo var formulario1 = document.forms['formularioContacto']; nos permite acceder a un formulario cuyo atributo name es formularioContacto.

Finalmente, un elemento de un formulario tiene una propiedad que es una referencia al formulario en el cual está inserto. La sintaxis sería del tipo:

```
var formulario1 = document.getElementById['valorIdDelCampo'].form;
```

Suponiendo que un formulario tiene un campo como: <label>Apellidos:<input id="apellidosFormContacto" type="text" name="apellidos" /></label>, podemos acceder al formulario usando document.getElementById('apellidosFormContacto').form

16.2 UTILIDADES BÁSICAS PARA FORMULARIOS

16.2.1 OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO

La mayoría de técnicas JavaScript relacionadas con los formularios requieren leer y/o modificar el valor de los campos del formulario. Por tanto, a continuación se muestra cómo obtener el valor de los campos de formulario más utilizados.

16.2.1.1 CUADRO DE TEXTO Y TEXTAREA

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

```
undefined - Untitled-1
1  <input type="text" id="texto" />
2  var valor = document.getElementById("texto").value;
3  <textarea id="parrafo"></textarea>
4  var valor = document.getElementById("parrafo").value;
```

16.2.1.2 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*:

```
undefined - Untitled-1
1  <input type="radio" value="si" name="pregunta" id="pregunta_si"/> SI
2  <input type="radio" value="no" name="pregunta" id="pregunta_no"/> NO
3  <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:

16.2.1.3 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:

```
undefined - Untitled-1
1  var elementos = document.getElementsByName("pregunta");
2
3  for(var i=0; i<elementos.length; i++) {
4    console.log(" Elemento: " + elementos[i].value + "\n Seleccionado: " + ele
   mentos[i].checked);
5  }
6
7  <input type="checkbox" value="condiciones" name="condiciones" id="condicione
   s"/> He leído y acepto las condiciones
8  <input type="checkbox" value="privacidad" name="privacidad" id="privacidad"/
   > He leído la política de privacidad
```

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

```
undefined - Untitled-1
1  var elemento = document.getElementById("condiciones");
2  console.log(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.
   checked);
3
4  elemento = document.getElementById("privacidad");
5  console.log(" Elemento: " + elemento.value + "\n Seleccionado: " + elemento.
   checked);
6
```

16.2.1.4 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:

```
// 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;

console.log("Opción seleccionada: " + textoSeleccionado + "\n Valor de la opción: " + valorSeleccionado);
```

- 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.

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:

undefined - Untitled-1

```
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).

16.2.2 ESTABLECER EL FOCO EN UN ELEMENTO

En programación, cuando un elemento está seleccionado y se puede escribir directamente en el o se puede modificar alguna de sus propiedades, se dice que tiene el foco del programa.

Si un cuadro de texto de un formulario tiene el foco, el usuario puede escribir directamente en el sin necesidad de pinchar previamente con el ratón en el interior del cuadro. Igualmente, si una lista desplegable tiene el foco, el usuario puede seleccionar una opción directamente subiendo y bajando con las flechas del teclado.

Al pulsar repetidamente la tecla TABULADOR sobre una página web, los diferentes elementos (enlaces, imágenes, campos de formulario, etc.) van obteniendo el foco del navegador (el elemento seleccionado cada vez suele mostrar un pequeño borde punteado).

Si en una página web el formulario es el elemento más importante, como por ejemplo en una página de búsqueda o en una página con un formulario para registrarse, se considera una buena práctica de usabilidad el asignar automáticamente el foco al primer elemento del formulario cuando se carga la página.

Para asignar el foco a un elemento de XHTML, se utiliza la función focus(). El siguiente ejemplo asigna el foco a un elemento de formulario cuyo atributo id es igual a primero:

undefined - Untitled-1

```
document.getElementById("primero").focus();
<form id="formulario" action="#">
  <input type="text" id="primero" />
</form>

```

Ampliando el ejemplo anterior, se puede asignar automáticamente el foco del programa al primer elemento del primer formulario de la página, independientemente del id del formulario y de los elementos:

undefined - Untitled-1

```
if(document.forms.length > 0) {
    if(document.forms[0].elements.length > 0) {
      document.forms[0].elements[0].focus();
    }
  }

```

El código anterior comprueba que existe al menos un formulario en la página mediante el tamaño del array forms. Si su tamaño es mayor que 0, se utiliza este primer formulario. Empleando la misma técnica, se comprueba que el formulario tenga al menos un elemento (if(document.forms[0].elements.length > 0)). En caso afirmativo, se establece el foco del navegador en el primer elemento del primer formulario (document.forms[0].elements[0].focus();).

Para que el ejemplo anterior sea completamente correcto, se debe añadir una comprobación adicional. El campo de formulario que se selecciona no debería ser de tipo hidden:

```
undefined - Untitled-1
1  if(document.forms.length > 0) {
2  for(var i=0; i < document.forms[0].elements.length; i++) {
3      var campo = document.forms[0].elements[i];
4      if(campo.type != "hidden") {
5      campo.focus();
6      break;
7      }
8  }
9  }
```

16.2.3 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:

```
undefined - Untitled-1
1  <form id="formulario" action="#">
2    ...
3    <input type="button" value="Enviar" onclick="this.disabled=true; this.valu
   e='Enviando...'; this.form.submit()" />
4  </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 tiposubmit`, el botón se deshabilita antes de enviar el formulario y por tanto el formulario acaba sin enviarse.

16.2.4 LIMITAR EL TAMAÑO DE CARACTERES DE UN TEXTAREA

La carencia más importante de los campos de formulario de tipo textarea es la imposibilidad de limitar el máximo número de caracteres que se pueden introducir, de forma similar al atributo maxlength de los cuadros de texto normales.

JavaScript permite añadir esta característica de forma muy sencilla. En primer lugar, hay que recordar que con algunos eventos (como onkeypress, onclick y onsubmit) se puede evitar su comportamiento normal si se devuelve el valor false.

Evitar el comportamiento normal equivale a modificar completamente el comportamiento habitual del evento. Si por ejemplo se devuelve el valor false en el evento onkeypress, la tecla pulsada por el usuario no se tiene en cuenta. Si se devuelve false en el evento onclick de un elemento como un enlace, el navegador no carga la página indicada por el enlace.

Si un evento devuelve el valor true, su comportamiento es el habitual:

```
<textarea onkeypress="return true;"></textarea>
```

En el textarea del ejemplo anterior, el usuario puede escribir cualquier carácter, ya que el evento onkeypress devuelve true y por tanto, su comportamiento es el normal y la tecla pulsada se transforma en un carácter dentro del textarea.

Sin embargo, en el siguiente ejemplo:

```
<textarea onkeypress="return false;"></textarea>
```

Como el valor devuelto por el evento onkeypress es igual a false, el navegador no ejecuta el comportamiento por defecto del evento, es decir, la tecla presionada no se transforma en ningún carácter dentro del textarea. No importa las veces que se pulsen las teclas y no importa la tecla pulsada, ese textarea no permitirá escribir ningún carácter.

Aprovechando esta característica, es sencillo limitar el número de caracteres que se pueden escribir en un elemento de tipo textarea: se comprueba si se ha llegado al máximo número de caracteres permitido y en caso afirmativo se evita el comportamiento habitual del evento y por tanto, los caracteres adicionales no se añaden al textarea:

undefined - Untitled-1

```
function limita(maximoCaracteres) {
    var elemento = document.getElementById("texto");
    if(elemento.value.length >= maximoCaracteres ) {
      return false;
    }
    else {
      return true;
    }
  }
  <textarea id="texto" onkeypress="return limita(100);"></textarea>
```

En el ejemplo anterior, con cada tecla pulsada se compara el número total de caracteres del textarea con el máximo número de caracteres permitido. Si el número de caracteres es igual o mayor que el límite, se devuelve el valor false y por tanto, se evita el comportamiento por defecto de onkeypress y la tecla no se añade.

16.2.5 RESTRINGIR LOS CARACTERES PERMITIDOS EN UN CUADRO DE TEXTO

En ocasiones, puede ser útil bloquear algunos caracteres determinados en un cuadro de texto. Si por ejemplo un cuadro de texto espera que se introduzca un número, puede ser interesante no permitir al usuario introducir ningún carácter que no sea numérico.

Igualmente, en algunos casos puede ser útil impedir que el usuario introduzca números en un cuadro de texto. Utilizando el evento onkeypress y unas cuantas sentencias JavaScript, el problema se resuelve fácilmente:

undefined - Untitled-1

```
function permite(elEvento, permitidos) {
  // Variables que definen los caracteres permitidos
  var numeros = "0123456789";
  var caracteres = " abcdefghijklmnñopqrstuvwxyzABCDEFGHIJKLMNÑOPQRSTUVWXYZ";
  var numeros_caracteres = numeros + caracteres;
  var teclas_especiales = [8, 37, 39, 46];
  // 8 = BackSpace, 46 = Supr, 37 = flecha izquierda, 39 = flecha derecha

  // Seleccionar los caracteres a partir del parámetro de la función
  switch(permitidos) {
    case 'num':
      permitidos = numeros;
      break;
    case 'car':
      permitidos = caracteres;
      break;
    case 'num_car':
      permitidos = numeros_caracteres;
      break;
  }

  // Obtener la tecla pulsada
  var evento = elEvento || window.event;
  var codigoCaracter = evento.charCode || evento.keyCode;
  var caracter = String.fromCharCode(codigoCaracter);

  // Comprobar si la tecla pulsada es alguna de las teclas especiales
  // (teclas de borrado y flechas horizontales)
  var tecla_especial = false;
  for(var i in teclas_especiales) {
    if(codigoCaracter == teclas_especiales[i]) {
      tecla_especial = true;
      break;
    }
  }

  // Comprobar si la tecla pulsada se encuentra en los caracteres permitidos
  // o si es una tecla especial
  return permitidos.indexOf(caracter) != -1 || tecla_especial;
}
```

El funcionamiento del script anterior se basa en permitir o impedir el comportamiento habitual del evento onkeypress. Cuando se pulsa una tecla, se comprueba si el carácter de esa tecla se encuentra dentro de los caracteres permitidos para ese elemento <input>.

Si el carácter se encuentra dentro de los caracteres permitidos, se devuelve true y por tanto el comportamiento de onkeypress es el habitual y la tecla se escribe. Si el carácter no se encuentra dentro de los caracteres permitidos, se devuelve false y por tanto se impide el comportamiento normal de onkeypress y la tecla no llega a escribirse en el input.

Además, el script anterior siempre permite la pulsación de algunas teclas *especiales*. En concreto, las teclas BackSpace y Supr para borrar caracteres y las teclas Flecha Izquierda y Flecha Derecha para moverse en el cuadro de texto siempre se pueden pulsar independientemente del tipo de caracteres permitidos.

16.3 VALIDACIÓN

introducidos por los usuarios. Antes de enviar un formulario al servidor, se recomienda validar mediante JavaScript los datos insertados por el usuario. De esta forma, si el usuario ha cometido algún error al rellenar el formulario, se le puede notificar de forma instantánea, sin necesidad de esperar la respuesta del servidor.

Notificar los errores de forma inmediata mediante JavaScript mejora la satisfacción del usuario con la aplicación (lo que técnicamente se conoce como "mejorar la experiencia de usuario") y ayuda a reducir la carga de procesamiento en el servidor.

Normalmente, la validación de un formulario consiste en llamar a una función de validación cuando el usuario pulsa sobre el botón de envío del formulario. En esta función, se comprueban si los valores que ha introducido el usuario cumplen las restricciones impuestas por la aplicación.

Aunque existen tantas posibles comprobaciones como elementos de formulario diferentes, algunas comprobaciones son muy habituales: que se rellene un campo obligatorio, que se seleccione el valor de una lista desplegable, que la dirección de email indicada sea correcta, que la fecha introducida sea lógica, que se haya introducido un número donde así se requiere, etc.

A continuación se muestra el código JavaScript básico necesario para incorporar la validación a un formulario:

undefined - Untitled-1

```
function validacion() {
    if (condicion que debe cumplir el primer campo del formulario) {
      // Si no se cumple la condicion...
      console.log('[ERROR] El campo debe tener un valor de...');
      return false;
    }
    else if (condicion que debe cumplir el segundo campo del formulario) {
      // Si no se cumple la condicion...
      console.log('[ERROR] El campo debe tener un valor de...');
      return false;
    }else if (condicion que debe cumplir el último campo del formulario) {
      // Si no se cumple la condicion...
      console.log('[ERROR] El campo debe tener un valor de...');
      return false;
    }

    // Si el script ha llegado a este punto, todas las condiciones
    // se han cumplido, por lo que se devuelve el valor true
    return true;
  }
```

El funcionamiento de esta técnica de validación se basa en el comportamiento del evento onsubmit de JavaScript. Al igual que otros eventos como onclick y onkeypress, el evento 'onsubmit varía su comportamiento en función del valor que se devuelve.

Así, si el evento onsubmit devuelve el valor true, el formulario se envía como lo haría normalmente. Sin embargo, si el evento onsubmit devuelve el valor false, el formulario no se envía. La clave de esta técnica consiste en comprobar todos y cada uno de los elementos del formulario. En cuando se encuentra un elemento incorrecto, se devuelve el valor false. Si no se encuentra ningún error, se devuelve el valor true.

Por lo tanto, en primer lugar se define el evento onsubmit del formulario como:

```
onsubmit="return validacion()"
```

Como el código JavaScript devuelve el valor resultante de la función validacion(), el formulario solamente se enviará al servidor si esa función devuelve true. En el caso de que la función validacion() devuelva false, el formulario permanecerá sin enviarse.

Dentro de la función validacion() se comprueban todas las condiciones impuestas por la aplicación. Cuando no se cumple una condición, se devuelve false y por tanto el formulario no se envía. Si se llega al final de la función, todas las condiciones se han cumplido correctamente, por lo que se devuelve true y el formulario se envía.

La notificación de los errores cometidos depende del diseño de cada aplicación. En el código del ejemplo anterior simplemente se muestran mensajes mediante la función console.log() indicando el error producido. Las aplicaciones web mejor diseñadas muestran cada mensaje de error al lado del elemento de formulario correspondiente y también suelen mostrar un mensaje principal indicando que el formulario contiene errores.

Una vez definido el esquema de la función validacion(), se debe añadir a esta función el código correspondiente a todas las comprobaciones que se realizan sobre los elementos del formulario. A continuación, se muestran algunas de las validaciones más habituales de los campos de formulario.

16.3.1 VALIDAR UN CAMPO DE TEXTO OBLIGATORIO

Se trata de forzar al usuario a introducir un valor en un cuadro de texto o textarea en los que sea obligatorio. La condición en JavaScript se puede indicar como:

Para que se de por completado un campo de texto obligatorio, se comprueba que el valor introducido sea válido, que el número de caracteres introducido sea mayor que cero y que no se hayan introducido sólo espacios en blanco.

La palabra reservada null es un valor especial que se utiliza para indicar *"ningún valor"*. Si el valor de una variable es null, la variable no contiene ningún valor de tipo objeto, array, numérico, cadena de texto o booleano.

La segunda parte de la condición obliga a que el texto introducido tenga una longitud superior a cero caracteres, esto es, que no sea un texto vacío.

Por último, la tercera parte de la condición (/^\s+$/.test(valor)) obliga a que el valor introducido por el usuario no sólo esté formado por espacios en blanco. Esta comprobación se basa en el uso de "expresiones regulares", un recurso habitual en cualquier lenguaje de programación pero que por su gran complejidad no se van a estudiar. Por lo tanto, sólo es necesario copiar literalmente esta condición, poniendo especial cuidado en no modificar ningún carácter de la expresión.

16.3.2 VALIDAR UN CAMPO DE TEXTO CON VALORES NUMÉRICOS

Se trata de obligar al usuario a introducir un valor numérico en un cuadro de texto. La condición JavaScript consiste en:

undefined - Untitled-1

```
  valor = document.getElementById("campo").value;
if( isNaN(valor) ) {
  return false;
}
```

Si el contenido de la variable valor no es un número válido, no se cumple la condición. La ventaja de utilizar la función interna isNaN() es que simplifica las comprobaciones, ya que JavaScript se encarga de tener en cuenta los decimales, signos, etc.

A continuación se muestran algunos resultados de la función isNaN():

undefined - Untitled-1

```
isNaN(3);          // false
isNaN("3");        // false
isNaN(3.3545);     // false
isNaN(32323.345);  // false
isNaN(+23.2);      // false
isNaN("-23.2");    // false
isNaN("23a");      // true
isNaN("23.43.54"); // true
```

16.3.3 VALIDAR QUE SE HA SELECCIONADO UNA OPCIÓN DE UNA LISTA

Se trata de obligar al usuario a seleccionar un elemento de una lista desplegable. El siguiente código JavaScript permite conseguirlo:

```
undefined - Untitled-1

indice = document.getElementById("opciones").selectedIndex;
if( indice == null || indice == 0 ) {
  return false;
}
<select id="opciones" name="opciones">
  <option value="">- Selecciona un valor -</option>
  <option value="1">Primer valor</option>
  <option value="2">Segundo valor</option>
  <option value="3">Tercer valor</option>
</select>
```

A partir de la propiedad selectedIndex, se comprueba si el índice de la opción seleccionada es válido y además es distinto de cero. La primera opción de la lista (- Selecciona un valor –) no es válida, por lo que no se permite el valor 0 para esta propiedad selectedIndex.

16.3.4 VALIDAR UNA DIRECCIÓN DE EMAIL

Se trata de obligar al usuario a introducir una dirección de email con un formato válido. Por tanto, lo que se comprueba es que la dirección parezca válida, ya que no se comprueba si se trata de una cuenta de correo electrónico real y operativa. La condición JavaScript consiste en:

```
undefined - Untitled-1

valor = document.getElementById("campo").value;
if( !(/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)/.test(valor)) ) {
  return false;
}
```

La comprobación se realiza nuevamente mediante las expresiones regulares, ya que las direcciones de correo electrónico válidas pueden ser muy diferentes. Por otra parte, como el estándar que define el formato de las direcciones de correo electrónico es muy complejo, la expresión regular anterior es una simplificación. Aunque esta regla valida la mayoría de direcciones de correo electrónico utilizadas por los usuarios, no soporta todos los diferentes formatos válidos de email.

16.3.5 VALIDAR UNA FECHA

Las fechas suelen ser los campos de formulario más complicados de validar por la multitud de formas diferentes en las que se pueden introducir. El siguiente código asume que de alguna forma se ha obtenido el año, el mes y el día introducidos por el usuario:

La función Date(ano, mes, dia) es una función interna de JavaScript que permite construir fechas a partir del año, el mes y el día de la fecha. Es muy importante tener en cuenta que el número de mes se indica de 0 a 11, siendo 0 el mes de Enero y 11 el mes de Diciembre. Los días del mes siguen una numeración diferente, ya que el mínimo permitido es 1 y el máximo 31.

La validación consiste en intentar construir una fecha con los datos proporcionados por el usuario. Si los datos del usuario no son correctos, la fecha no se puede construir correctamente y por tanto la validación del formulario no será correcta.

16.3.6 VALIDAR UN NÚMERO DE DNI

Se trata de comprobar que el número proporcionado por el usuario se corresponde con un número válido de Documento Nacional de Identidad o DNI. Aunque para cada país o región los requisitos del documento de identidad de las personas pueden variar, a continuación se muestra un ejemplo genérico fácilmente adaptable. La validación no sólo debe comprobar que

el número esté formado por ocho cifras y una letra, sino que también es necesario comprobar que la letra indicada es correcta para el número introducido:

undefined - Untitled-1

```
valor = document.getElementById("campo").value;
var letras = ['T', 'R', 'W', 'A', 'G', 'M', 'Y', 'F', 'P', 'D', 'X', 'B',
'N', 'J', 'Z', 'S', 'Q', 'V', 'H', 'L', 'C', 'K', 'E', 'T'];

if( !(/^\d{8}[A-Z]$/.test(valor)) ) {
  return false;
}

if(valor.charAt(8) != letras[(valor.substring(0, 8))%23]) {
  return false;
}
```

La primera comprobación asegura que el formato del número introducido es el correcto, es decir, que está formado por 8 números seguidos y una letra. Si la letra está al principio de los números, la comprobación sería /^[A-Z]\d{8}$/. Si en vez de ocho números y una letra, se requieren diez números y dos letras, la comprobación sería /^\d{10}[A-Z]{2}$/ y así sucesivamente.

La segunda comprobación aplica el algoritmo de cálculo de la letra del DNI y la compara con la letra proporcionada por el usuario. El algoritmo de cada documento de identificación es diferente, por lo que esta parte de la validación se debe adaptar convenientemente.

16.3.9 VALIDAR QUE UN RADIOBUTTON HA SIDO SELECCIONADO

Aunque se trata de un caso similar al de los *checkbox*, la validación de los *radiobutton* presenta una diferencia importante: en general, la comprobación que se realiza es que el usuario haya seleccionado algún *radiobutton* de los que forman un determinado grupo. Mediante JavaScript, es sencillo determinar si se ha seleccionado algún *radiobutton* de un grupo:

El anterior ejemplo recorre todos los radiobutton que forman un grupo y comprueba elemento por elemento si ha sido seleccionado. Cuando se encuentra el primer radiobutton seleccionado, se sale del bucle y se indica que al menos uno ha sido seleccionado.

FUNCIÓN DOCUMENT.WRITE

Uno de los comandos básicos de JavaScript es document.write. Esto imprime el texto especificado en la página. Para imprimir texto literalmente, escribe el texto en comillas simples y entre paréntesis como sigue:

```
document.write('Hola Mundo!');
```

El código anterior imprimira el mensaje "Hola Mundo!" que aparecen en la página.

También puedes utilizar document.write para imprimir variables. Introduce el nombre de la variable sin comillas, como por ejemplo:

```
 var mytext = "Hola otra vez";
document.write(mytext);
```

Ten en cuenta que si se colocas comillas alrededor del nombre de la variable, se imprimirá el nombre de la variable (en lugar del valor de la variable). También puedes combinar valores de variables y cadenas de texto mediante el signo +:

```
var colour1 = "morado";
var colour2 = "rosa";
document.write('<p>colour1: ' + colour1 + '<br>colour2: ' + colour2 + '</p>');
```

Muestra los nombres de las variables, así como los valores. Mostrará lo siguiente:

colour1: púrpura
colour2: Rosa

Recuerda, si le pones comillas al texto se imprimirá el texto literalmente. Utiliza el texto sin comillas solo cuando quieras imprimir el valor de una variable.

PROMPT

Para la entrada de datos por teclado tenemos el método prompt. Cada vez que necesitamos ingresar un dato con este método aparece una ventana donde cargamos el valor. Hay otras formas más sofisticadas para la entrada de datos en una página HTML, pero para el aprendizaje de los conceptos básicos de JavaScript nos resultará más práctica esta metodología (luego aprenderemos a recuperar datos de un formulario HTML).

Para ver su funcionamiento analicemos este ejemplo:

undefined - Untitled-2

```
<!DOCTYPE html>
<html>

<head>
    <title>Ejemplo de JavaScript</title>
    <meta charset="UTF-8">
</head>

<body>

    <script>
        let nombre;
        let edad;
        nombre = prompt('Ingrese su nombre:');
        edad = prompt('Ingrese su edad:');
        document.write('Hola ');
        document.write(nombre);
        document.write(' asi que tienes ');
        document.write(edad);
        document.write(' años');
    </script>

</body>

</html>
```