JAVASCRIPT
Capítulo 7

Lenguajes de Marcas y Sistemas de Gestión de Información

Curso 2019/2020

Capítulo 7: FORMULARIOS

1. Introducción
2. Propiedades Básicas de Formularios y Elementos
3. Utilidades Básicas para Formularios
4. Validaciones

1. INTRODUCCIÓN

La programación con formularios web es una de las tareas fundamentales de JavaScript → Una de las principales **razones** por las que se inventó JavaScript fue la necesidad de validar los datos de los formularios directamente en el navegador del usuario, **evitando recargar** la página cuando el usuario cometía errores al rellenar los formularios.

La aparición de AJAX **ha relevado** al tratamiento de formularios como la principal actividad de JavaScript. Ahora, el **principal uso** de JavaScript es el de las comunicaciones asíncronas con los servidores y el de la manipulación dinámica de las aplicaciones.

2. PROPIEDADES BÁSICAS DE FORMULARIOS Y ELEMENTOS

Cuando se carga una página web, el navegador crea automáticamente un array llamado `forms` y que contiene la referencia a **todos los formularios** de la página.

Para acceder al array `forms`, se utiliza el objeto `document`, por lo que `document.forms` es el array que contiene todos los formularios de la página. El acceso a cada formulario se realiza con la misma sintaxis de los arrays:

```
document.forms[0];
```

El navegador **crea automáticamente** un array llamado `elements` por cada uno de los formularios de la página. Cada array `elements` contiene la referencia a todos los elementos de ese formulario:

```
document.forms[0].elements[0];
```

La sintaxis de los arrays **no es tan concisa**. El ejemplo muestra cómo obtener el último elemento del primer formulario de la página:

```
document.forms[0].elements[document.forms[0].elements.length-1];
```

2. PROPIEDADES BÁSICAS DE FORMULARIOS Y ELEMENTOS

¿Qué sucede si cambia el diseño de la página y en el código HTML se cambia el orden de los formularios originales o se añaden nuevos formularios?

Es difícil confiar en que el orden de los formularios se mantenga → Siempre **debería evitarse** el acceso a los formularios de una página mediante el array `document.forms`.

Para evitar los problemas del método anterior se puede acceder a los formularios a través de su nombre (`name`) o a través de su atributo `id`:

```
var formularioPrincipal = document.formulario;
var formularioSecundario = document.otro_formulario;
```

```
 <form name="formulario" >
   ...
</form>
 <form name="otro_formulario" >
   ...
</form>
```

2. PROPIEDADES BÁSICAS DE FORMULARIOS Y ELEMENTOS

También se puede acceder a los formularios y a sus elementos utilizando las **funciones DOM** de acceso directo a los nodos. El siguiente ejemplo utiliza la función `document.getElementById()` para acceder de forma directa a un formulario y a uno de sus elementos:

```
var formularioPrincipal = document.getElementById("formulario");

var primerElemento = document.getElementById("elemento");
```

```
<form name="formulario" id="formulario" >

  <input type="text" name="elemento" id="elemento" />

</form>
```

2. PROPIEDADES BÁSICAS DE FORMULARIOS Y ELEMENTOS: *PROPIEDADES*

- **type**: tipo de elemento que se trata. Para los elementos de tipo `<input>` coincide con el valor de su atributo `type`. Para las listas desplegables normales (elemento `<select>`) su valor es `select-one`, lo que permite diferenciarlas de las listas que permiten seleccionar varios elementos a la vez y cuyo tipo es `select-multiple`.
- **form**: referencia directa al formulario al que pertenece el elemento. Para acceder al formulario de un elemento, se utiliza `document.getElementById("id_elemento").form`
- **name**: obtiene el valor del atributo `name` de XHTML. Solamente se puede leer su valor, por lo que no se puede modificar.
- **value**: permite leer y modificar el valor del atributo `value` de XHTML. Para los campos de texto obtiene el texto que ha escrito el usuario. Para los botones obtiene el texto que se muestra en el botón.

2. PROPIEDADES BÁSICAS DE FORMULARIOS Y ELEMENTOS: *EVENTOS*

- **`onclick`**: se produce cuando se pincha con el ratón sobre un elemento. Se utiliza con cualquiera de los tipos de botones que permite definir XHTML (`<input type="button">`, `<input type="submit">`, `<input type="image">`).
- **`onchange`**: se produce cuando el usuario cambia el valor de un elemento de texto. También se produce cuando el usuario selecciona una opción en una lista desplegable (`<select>`). El evento sólo se produce si **después de realizar el cambio**, el usuario pasa al siguiente campo del formulario, lo que técnicamente se conoce como que "el otro campo de formulario ha perdido el foco".
- **`onfocus`**: se produce cuando el usuario selecciona un elemento del formulario.
- **`onblur`**: evento complementario de `onfocus`, ya que se produce cuando el usuario ha deseleccionado un elemento por haber seleccionado otro elemento del formulario. Técnicamente, se dice que el elemento anterior "ha perdido el foco".

3. UTILIDADES BÁSICAS PARA FORMULARIOS

- Obtener el **Valor** de los Campos de Formulario.
- Establecer el **Foco** en un Elemento.
- Evitar el **Envío Duplicado** de un Formulario.
- **Limitar** el Tamaño de Caracteres de un `textarea`.
- **Restringir** los Caracteres permitidos a un Cuadro de Texto.

3.1. 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:

- **Cuadro de texto o** `textarea`
- `radiobutton`
- `checkbox`
- `select`

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: `text` ó `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;
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: radiobutton

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` si está seleccionado y `false` en cualquier otro caso:

```
<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
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: radiobutton

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);
}
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: checkbox

Son muy similares a los radiobutton, salvo que en este caso se debe comprobar cada checkbox **de forma independiente** del resto.

Los grupos de radiobutton son **mutuamente excluyentes** y sólo se puede seleccionar **uno de ellos**, mientras que los checkbox se pueden seleccionar de forma independiente respecto de los demás:

```
<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 política de privacidad
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: checkbox

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);
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: `select`

Las listas desplegables (`<select>`) son los elementos en los que es más difícil obtener su valor:

```
<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>
```

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: `select`

Se pretende obtener el valor del atributo `value` de la opción seleccionada. Para obtener el valor seleccionado, deben utilizarse las siguientes propiedades:

- **`options`**: array creado automáticamente por el navegador para cada lista desplegable y que contiene la **referencia** a todas las opciones de esa lista. Ej: 1ª 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 por el navegador para cada lista.

3.1. OBTENER EL VALOR DE LOS CAMPOS DE FORMULARIO: `select`

Para obtener el valor del atributo `value` correspondiente a la opción seleccionada por el usuario:

```
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.

3.2. ESTABLECER EL FOCO DE UN ELEMENTO

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.

Si pulsas el TABULADOR sobre una página web, los diferentes elementos (enlaces, imágenes, campos de formulario, etc.) van **obteniendo el foco** del navegador.

Si en una página web el formulario es el elemento más importante, se considera una buena práctica de usabilidad el **asignar automáticamente** el **foco** al **primer elemento del formulario** cuando se carga la página.

3.2. ESTABLECER EL FOCO DE UN ELEMENTO

Para asignar el foco a un elemento de XHTML, se utiliza la función `focus()`:

```
document.getElementById("primero").focus();
```

```
<form id="formulario" action="#">
  <input type="text" id="primero" />
</form>
```

Si queremos **asignar automáticamente el foco del programa al primer elemento del primer formulario de la página:**

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

Comprueba que **existe al menos un formulario** en la página; Se comprueba que el **formulario tenga** al menos un **elemento**; En caso afirmativo, se establece el **foco** del navegador en el **primer elemento** del primer formulario.

3.2. ESTABLECER EL FOCO DE UN ELEMENTO

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

```
if(document.forms.length > 0) {
  for(var i=0; i < document.forms[0].elements.length; i++) {
    var campo = document.forms[0].elements[i];
    if(campo.type != "hidden") {
      campo.focus();
      break;
    }
  }
}
```

3.3. EVITAR EL ENVÍO DUPLICADO DE UN FORMULARIO

Uno de los problemas habituales de formularios web es la posibilidad de que el usuario **pulse dos veces** seguidas sobre el botón "Enviar".

Una buena práctica en el diseño de aplicaciones web suele ser la de **deshabilitar el botón** de envío después de la primera pulsación:

```
<form id="formulario" action="#">

  ...

  <input type="button" value="Enviar"
onclick="this.disabled=true; this.value='Enviando...';
this.form.submit()" />

</form>
```

3.3. EVITAR EL ENVÍO DUPLICADO DE UN FORMULARIO

Cuando se pulsa el botón de envío, se produce el evento `onclick` sobre el botón, ejecutándose 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 a `"Enviando..."`
3. Por último, **se envía el formulario** mediante la función `submit()` en la siguiente instrucción: `this.form.submit()`

3.4. LIMITAR EL TAMAÑO DE CARACTERES DE UN `textarea`

En los campos de formulario de tipo `textarea` es **imposible limitar el máximo número de caracteres** que se pueden introducir.

Con algunos eventos (`onkeypress`, `onclick` y `onsubmit`) se puede evitar su **comportamiento normal** si se devuelve el valor `false`: equivale a modificar el comportamiento habitual del evento. Si un evento devuelve el valor `true`, su comportamiento es el habitual:

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

El usuario puede escribir cualquier carácter, ya que el evento `onkeypress` devuelve `true`. Su comportamiento es el normal y la tecla pulsada se transforma en un carácter dentro del `textarea`.

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

El valor devuelto por `onkeypress` es `false`. El navegador **no ejecuta** el comportamiento por defecto del evento: la tecla presionada no se transforma en ningún carácter dentro del `textarea`. **Ese `textarea` no permitirá escribir ningún carácter.**

3.4. LIMITAR EL TAMAÑO DE CARACTERES DE UN `textarea`

SOLUCIÓN: 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`:

```
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>
```

Con cada tecla pulsada se compara el nº total de caracteres del `textarea` con el máximo nº de caracteres permitido. Si el nº 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.

3.5. RESTRINGIR LOS CARACTERES PERMITIDOS A UN CUADRO DE TEXTO

Puede ser útil **bloquear algunos caracteres** determinados en un cuadro de texto. Si un cuadro de texto espera que se introduzca un número, puede ser interesante no permitir introducir ningún carácter que no sea numérico o en algunos casos puede ser útil impedir que el usuario introduzca números en un cuadro de texto.

El funcionamiento del script 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 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 se impide el comportamiento normal de `onkeypress` y la tecla no se escribe en el `<input>`.

EJEMPLO

4. VALIDACIONES

La principal utilidad de JS en el manejo de formularios es la **validación de datos introducidos** por usuarios. Antes de enviar un formulario, se recomienda validar mediante JS los datos insertados → 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 JS mejora la **satisfacción del usuario** con la aplicación y ayuda a **reducir la carga** de procesamiento en el servidor.

La validación de un formulario consiste en llamar a una **función de validación** cuando el usuario pulsa sobre el botón de envío del formulario donde se comprueban si los valores que se han introducido cumplen las restricciones impuestas por la aplicación.

4. VALIDACIONES

Existen tantas **comprobaciones como elementos** de formulario diferentes: que se rellene un campo obligatorio, que se seleccione el valor de una lista desplegable, que la dirección de email sea correcta, que la fecha sea lógica, etc…

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

```
<form action="" method="" id="" name="" onsubmit="return validacion()">
   ...
</form>
```

4. VALIDACIONES: function validacion()

```
function validacion() {
  if (condicion que debe cumplir el primer campo del formulario) {
    // Si no se cumple la condicion...
    alert('[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...
    alert('[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...
    alert('[ERROR] El campo debe tener un valor de...');
    return false;
  }
  //Si llega aqui, todas las condiciones se han cumplido y se devuelve el valor true
  return true;
}
```

4. VALIDACIONES

El funcionamiento se basa en el comportamiento del evento `onsubmit` de JavaScript.

Si el evento devuelve el valor `true`, el formulario se envía. Sin embargo, si el evento 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`.

En primer lugar se define el evento `onsubmit` del formulario:

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

4. VALIDACIONES

El código JS devuelve el valor resultante de la función `validacion()`, el formulario solamente se enviará al servidor si esa función devuelve `true`. Si la función devuelve `false`, el formulario permanecerá sin enviarse.

Dentro de la función se comprueban **todas las condiciones** impuestas por la aplicación. 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** depende del diseño de cada aplicación. En el código ejemplo se muestran mensajes mediante la función `alert()` 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.

4.1. VALIDAR UN CAMPO DE TEXTO OBLIGATORIO

Forzar al usuario a introducir un valor en un cuadro de texto o `textarea` en los que sea obligatorio:

```
valor = document.getElementById("campo").value;

if( valor == null || valor.length == 0 || /^\s+$/.test(valor) ) {
  return false;
}
```

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**.

4.2. VALIDAR UN CAMPO DE TEXTO CON VALORES NUMÉRICOS

Se trata de obligar al usuario a introducir un valor numérico en un cuadro de texto:

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

Resultados de la función `isNaN()`**:**

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

4.3. VALIDAR QUE SE HA SELECCIONADO UNA OPCIÓN DE UNA LISTA

Se trata de obligar al usuario a seleccionar un elemento de una lista desplegable:

```
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.

4.4. VALIDAR UNA DIRECCIÓN DE UN EMAIL

Se trata de obligar al usuario a introducir una dirección de email con un formato válido.

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:

```
valor = document.getElementById("campo").value;

if(              !(/\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-
.]\w+)/.test(valor)) ) {

  return false;

}
```

4.5. 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:

```
var ano = document.getElementById("ano").value;

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

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

valor = new Date(ano, mes, dia);

if( !isNaN(valor) ) {

  return false;

}
```

La función `Date(ano, mes, dia)` es una función interna de JS 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.

4.6. VALIDAR UN NÚMERO DE DNI

Comprobar que el número proporcionado por el usuario se corresponde con un número válido de DNI:

```
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.
- La segunda comprobación **aplica el algoritmo de cálculo** de la letra del DNI y la compara con la letra proporcionada por el usuario.

4.7. VALIDAR UN NÚMERO DE TELÉFONO

Los números de teléfono pueden ser indicados de **formas muy diferentes**: con prefijo nacional, con prefijo internacional, agrupado por pares, separando los números con guiones, etc.

El siguiente script considera que un número de teléfono está formado por **nueve dígitos consecutivos** y sin espacios ni guiones entre las cifras:

```
valor = document.getElementById("campo").value;
if( !(/^\d{9}$/.test(valor)) ) {
      return false;
}
```

4.7. VALIDAR UN NÚMERO DE TELÉFONO

A continuación se muestran **otras expresiones regulares** que se pueden utilizar para otros formatos de número de teléfono:

Número	Expresión regular	Formato
900900900	/^\d{9}$/	9 cifras seguidas
900-900-900	/^\d{3}-\d{3}-\d{3}$/	9 cifras agrupadas de 3 en 3 y separadas por guiones
900 900 900	/^\d{3}\s\d{6}$/	9 cifras, las 3 primeras separadas por un espacio
900 90 09 00	/^\d{3}\s\d{2}\s\d{2}\s\d{2}$/	9 cifras, las 3 primeras separadas por un espacio, las siguientes agrupadas de 2 en 2
(900) 900900	/^\(\d{3}\)\s\d{6}$/	9 cifras, las 3 primeras encerradas por paréntesis y un espacio de separación respecto del resto
+34 900900900	/^\+\d{2,3}\s\d{9}$/	Prefijo internacional (+ seguido de 2 o 3 cifras), espacio en blanco y 9 cifras consecutivas

4.8. VALIDAR QUE UN `checkbox` HA SIDO SELECCIONADO

Si un elemento de tipo `checkbox` se debe seleccionar de forma obligatoria, JS permite comprobarlo de forma muy sencilla:

```
elemento = document.getElementById("campo");

if( !elemento.checked ) {

  return false;

}
```

4.8. VALIDAR QUE UN checkbox HA SIDO SELECCIONADO

Comprobar que todos los checkbox del formulario han sido seleccionados:

```
formulario = document.getElementById("formulario");
for(var i=0; i<formulario.elements.length; i++) {
        var elemento = formulario.elements[i];
        if(elemento.type == "checkbox") {
                if(!elemento.checked) {
                        return false;
                }
        }
}
```

4.9. VALIDAR QUE UN radiobutton HA SIDO SELECCIONADO

La comprobación que se realiza es que el usuario haya seleccionado algún `radiobutton`**:**

```
opciones = document.getElementsByName("opciones");
var seleccionado = false;
for(var i=0; i<opciones.length; i++) {
        if(opciones[i].checked) {
                seleccionado = true;
                break;
        }
}
if(!seleccionado) {
        return false;
}
```

Se recorren todos los `radiobutton` **y se 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.**