

Formularios en JavaScript

Módulo: **Desarrollo Web en entorno cliente**

CFGs Desarrollo de Aplicaciones Web

Introducción

- El uso de JavaScript en los formularios HTML se hace con el objetivo de validar los datos introducidos.
- Se hace en el cliente (navegador) para desligar de esta actividad al servidor que recibirá los datos introducidos por el usuario.
- El objeto principal es el FORM que contendrá los otros objetos: campos de texto, radiobutton, checkbox, textarea, etc.
- A través de eventos se realizarán los controles de validación. Por ejemplo, al cambiar de campo, al hacer el submit del formulario, etc.

Introducción

- Cuando se carga una página web el navegador crea automáticamente un array denominado “forms”. Por ejemplo, para acceder al primer formulario: `document.forms[0];`
- También se crea automáticamente un array “elements” que contiene los elementos del formulario. En este caso, para acceder al primer elemento del primer form: `document.forms[0].elements[0];`
- Inconveniente: En un entorno tan cambiante, es muy difícil confiar en que el orden de los formularios se mantenga estable en una página web. Por este motivo, **siempre debería evitarse el acceso a los formularios de una página mediante el array y acceder a través de su nombre o id.**

Propiedades y eventos del formulario

Algunas **propiedades** del formulario son:

- **name:** nombre del formulario.
- **action:** URL a la cual se envía el formulario.
- **method:** GET (cadena visible) y POST (cadena invisible).
- **target:** ventana o target donde se mostrarán los resultados.

Nota: Estas propiedades pueden ser modificadas desde JS cambiando el valor como respuesta a alguna acción del usuario.







Los **eventos** más utilizados serían: onSubmit y onReset.

```
<form name="nombre_formulario" action="procesar.php" method="POST"
target="_blank" onSubmit="alert('Se enviará el formulario')"
onReset="alert('Se borrarán todos los datos insertados')">
Escribe tu nombre: <input type="text" name="nombre"><br>
<input type="submit" value="enviar formulario">
<input type="reset" value="borrar">
</form>
```

Tipos de <input> de un formulario I

| Value | Description |
|----------------|---|
| button | Defines a clickable button (mostly used with a JavaScript to activate a script) |
| checkbox | Defines a checkbox |
| color |  Defines a color picker |
| date |  Defines a date control (year, month and day (no time)) |
| datetime |  The input type datetime has been removed from the HTML standard. Use datetime-local instead. |
| datetime-local |  Defines a date and time control (year, month, day, hour, minute, second, and fraction of a second (no time zone)) |
| email |  Defines a field for an e-mail address |
| file | Defines a file-select field and a "Browse..." button (for file uploads) |
| hidden | Defines a hidden input field |
| image | Defines an image as the submit button |
| month |  Defines a month and year control (no time zone) |
| number |  Defines a field for entering a number |

Tipos de <input> de un formulario II

| | | |
|----------|---|--|
| password | | Defines a password field (characters are masked) |
| radio | | Defines a radio button |
| range |  | Defines a control for entering a number whose exact value is not important (like a slider control) |
| reset | | Defines a reset button (resets all form values to default values) |
| search |  | Defines a text field for entering a search string |
| submit | | Defines a submit button |
| tel |  | Defines a field for entering a telephone number |
| text | | Default. Defines a single-line text field (default width is 20 characters) |
| time |  | Defines a control for entering a time (no time zone) |
| url |  | Defines a field for entering a URL |
| week |  | Defines a week and year control (no time zone) |

Propiedades y eventos de los elementos

Algunas **propiedades** útiles de los elementos de un formulario son:

- **type**: indica el tipo de elemento. Normalmente coincide con el valor del atributo type de los elementos <input>, salvo para los select que puede ser: select-one (listas normales) y select-multiple (permite la selección de más de un elemento).
- **form**: es una referencia al formulario al que pertenece el elemento.

Por ejemplo: `document.getElementById("campo").form`

- **value**: permite leer y modificar el valor del atributo value. En el caso de los botones obtiene el texto que se muestra en el botón.

Los **eventos** más utilizados serían: **onclick**, **onchange**, **onfocus** y **onblur**.

Obtener el valor de los campos

- **Cuadro de texto, textarea, etc.:**

```
var valor = document.getElementById("texto").value;
```

- **Radiobutton:**

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

- **Checkbox:**

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


Obtener el valor de los campos

- **Select:** En este caso, `options`, es un array con todas las opciones de la lista desplegable; `selectedIndex`, guarda el índice de la opción seleccionada.

```
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;
```

Establecer el foco

Focus(): función que asigna el foco a un elemento.

```
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;  
        }  
    }  
}
```

Evitar el reenvío de un formulario

Puede ocurrir que un usuario haga clic varias veces sobre el botón de “Enviar”. Para evitar esto se podría deshabilitar el botón, una vez se ha pulsado por primera vez:

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

Nota: si se utiliza un botón de tipo submit, el botón se deshabilita antes de enviar el formulario y el formulario no se envía.

Limitar tamaño de caracteres

- Input: usar la propiedad maxlength
- Textarea: usar manejadores de eventos que permitan la introducción o no de caracteres.

```
function limita(maximoCaracteres) {  
    var elemento = document.getElementById("texto");  
    if(elemento.value.length >= maximoCaracteres ) {  
        elemento.value = elemento.value.substring(0,maximoCaracteres);  
        return false;  
    } else {  
        return true;  
    }  
}  
  
<textarea id="texto" onChange="return limita(100);" onKeyUp="return  
limita(100);"></textarea>
```

Validar la extensión de un archivo I

```
<form method=post action="#" enctype="multipart/form-data">
<input type=file name="archivoupload">
<input type=button name="Submit" value="Enviar"
onclick="comprueba_extension(this.form, this.form.archivoupload.value)">
</form>
```

```
function comprueba_extension(formulario, archivo) {
    extensiones_permitidas = new Array(".gif", ".jpg", ".doc", ".pdf");
    mierror = "";
    if (!archivo) { //Si no tengo archivo
        mierror = "No has seleccionado ningún archivo";
    }else{ //recupero la extensión de este nombre de archivo
        extension = (archivo.substring(archivo.lastIndexOf("."))).toLowerCase();

        //compruebo si la extensión está entre las permitidas
        permitida = false;
        for (var i = 0; i < extensiones_permitidas.length; i++) {
            if (extensiones_permitidas[i] == extension) {
                permitida = true;
                break;
            }
        }
    }
}
```

Validar la extensión de un archivo II

```
if (!permitida) {  
    mierror = "Sólo se pueden subir archivos con extensiones: " +  
extensiones_permitidas.join();  
}else{  
  
    formulario.submit();  
    return 1;  
}  
}  
// no se ha podido submitir  
alert (mierror);  
return 0;  
}
```

Expresiones Regulares

- Una expresión regular es un patrón de caracteres que se puede utilizar para realizar búsquedas dentro de una cadena o un archivo.
- Las expresiones regulares se construyen de forma similar a las expresiones aritméticas, utilizando diversos operadores para combinar expresiones más sencillas.
- Las piezas fundamentales para la construcción de expresiones regulares son las que representan a un carácter simple. La mayoría de los caracteres, incluyendo las letras y los dígitos, se consideran expresiones regulares que se representan a sí mismos.
- Uso de las Expresiones Regulares es la validación de un formato específico en una cadena de caracteres dada, como por ejemplo fechas, correos electrónicos o identificadores.
- Cuando una expresión regular "encaja" con una cadena, línea o lo que sea se dice que coincide, o match (emparejar) en inglés.

Expresiones Regulares

En JS se construyen de dos formas:

- `var re = /ab+c/;` `//String`
- `var re = new RegExp("ab+c");` `//Constructor de objeto RegExp`

| Método | Descripción |
|------------------------------|---|
| <u>test</u> | Método del objeto RegExp que comprueba si empareja con un string. Devolverá true o false. |
| <u>match</u> | Método del objeto String que busca empareja en un string. Devolverá un array con la información del emparejamiento o null si no hay emparejamiento. |

Los argumentos de la expresión regular debe ir encerrada entre `/`
Por ejemplo., `/ex/` empareja la primera ocurrencia de `"ex"`

```
var regex = /[a-z]/;  
undefined  
var cadena = "Cesar Manrique";  
undefined  
cadena.match(regex);  
["e"]  
regex.test(cadena);  
true
```

Se distinguen
mayúsculas
de minúsculas

Usando los 2 métodos para
evaluar expresiones
regulares con JS

Expresiones Regulares - Opciones

Estos parámetros pueden usarse de manera separada o juntas en cualquier orden. Se incluyen en la propia expresión regular:

g (global): búsqueda global

Por ejemplo., `/ex/g` emparejaría con todas las ocurrencias de "ex"

i (ignore case): no es case sensitive

Por ejemplo., `/ex/i` emparejaría con las ocurrencias "ex", "EX", "Ex" y "eX"

```
"tom".match(/T/); // null
```

//Utilizando el parámetro i se pueden realizar búsquedas que no distingas mayúsculas y minúsculas.

```
"tom".match(/T/i); // ["t"]
```

```
"tom".match(/T/)
```

```
null
```

```
"tom".match(/T/i)
```

```
["t"]
```

Expresiones Regulares – Validar CP de 5 dígitos

```
<script>
  function checkpostal(){
    var re5digit=/^\d{5}$/
    if (document.myform.myinput.value.search(re5digit)==-1)
      alert("Introduce un número de 5 dígitos")
  }
</script>
<form name="myform">
  <input type="text" name="myinput" size=15>
  <input type="button" onClick="checkpostal()" value="check">
</form>
```

```
var re5digit=/^\d{5}$/
```

- ^ indica el comienzo de un string. Utilizando el metacaracter ^ indicamos que el emparejamiento o coincidencia debe empezar por el principio de la cadena.
- \d indica un dígito y el {5} a continuación significa que debe haber 5 dígitos consecutivos.
- \$ indica el final de cadena. Utilizando el metacaracter \$ obligamos que la coincidencia termine con el fin de la cadena.

Expresiones Regulares – Caracteres especiales: \

| Caracter | Significado |
|----------|--|
| \ | <p>Alguno de los siguientes: Para caracteres que generalmente son tratados como literales, indica que el siguiente carácter es un carácter especial y no debe ser interpretado como un literal. Por ejemplo, <code>/b/</code> coincide con el carácter 'b'. Sin embargo, colocando una barra invertida (backslash) antes de la b, usando así <code>/\b/</code>, el carácter se convierte en especial significando un límite de palabra.</p> <p>Para caracteres que generalmente son tratados como especiales, indica que el siguiente carácter no es especial y debería ser interpretado como un literal. Por ejemplo, <code>*</code> es un carácter especial que significa 0 o más ocurrencias del elemento que le precede; por ejemplo, <code>/a*/</code> significa cero o más a's. Para encontrar un <code>*</code> literalmente, hay que precederlo de una barra invertida; por ejemplo, <code>/a*/</code> coincide con 'a*'.</p> |

Expresiones Regulares – Caracteres especiales: ^

| Caracter | Significado |
|----------|------------------------------------|
| ^ | Empareja el comienzo de la entrada |

Por ejemplo,
/^U/ no coincide con la 'U' en "una U", pero sí coincide con la primera 'U' en "Una U".

Expresión regular: ^A

Emparejamiento: "Aaaann Animal Named Alex"

No empareja con: "an A"

Expresiones Regulares – Caracteres especiales: \$

| Caracter | Significado |
|----------|-----------------------------------|
| \$ | Empareja el final de la expresión |

Expresión regular: t\$

Emparejamiento: "Today I ate a tart"

No empareja con: "tart is here"

Expresiones Regulares – Caracteres especiales: *

| Caracter | Significado |
|----------|--|
| * | Empareja 0 o más veces el caracter que precede al * |

Expresión regular: `a*`

Emparejamiento: `"aaaaaaaaaaaaaaaaaallred"`

Emparejamiento: `"tart"`

No empareja con: `"tom"`

```
"aaaaaaaaaaaaaaaaaallred".match(/a*/g)
["aaaaaaaaaaaaaaaaa", "", "", "", "", "", ""]
"tart".match(/a*/g)
["", "a", "", "", ""]
"tom".match(/a*/g)
["", "", "", ""]
```

La última posición del array se corresponde con el caracter no visible de fin de línea

Expresión regular: `a*m`

Emparejamiento: `"tom"`

Expresiones Regulares – Caracteres especiales: +

| Caracter | Significado |
|----------|--|
| + | Empareja el caracter que lo precede 1 o más veces |

Expresión regular: 1+

Emparejamiento: "911"

Emparejamiento: "912"

No empareja con: "8675309"

Expresiones Regulares – Caracteres especiales: ?

| Caracter | Significado |
|----------|--|
| ? | Empareja el caracter que precede al ?, 0 o 1 vez |

Expresión regular: `l?`

Emparejamiento: `"lily"` (con la primera `"l"`)

Emparejamiento: `"llog"` (con la primera `"l"`)

No empareja con: `"Ron"`

```
"lily".match(/l?/)
```

```
["l"]
```

```
"llog".match(/l?/)
```

```
["l"]
```

```
"Ron".match(/l?/)
```

```
[""]
```


Expresiones Regulares – Caracteres especiales: .

| Caracter | Significado |
|----------|--|
| . | El caracter punto empareja con cualquier único caracter excepto el caracter que se encuentre en una línea nueva. |

Expresión regular: `.n`

Emparejamiento: "nay, **an** apple is **on** the tree"

No empareja con: "**nay**, an apple is on the tree"

```
"nay, an apple is on the tree".match(/.n/);  
["an"]
```

```
"nay, an apple is on the tree".match(/.n/g);  
["an", "on"]
```

Expresiones Regulares – Caracteres especiales: |

| Caracter | Significado |
|----------|---------------------------------|
| | Emparejamiento un patrón u otro |

x|y → Coincide con 'x' o 'y'. Por ejemplo, /green|red/ coincide con 'green' en "green apple" y 'red' en "red apple."

Expresión regular: red|blue

Emparejamiento: "hand me that blue crayon"

Emparejamiento: "hand me that red crayon"

No empareja con: "hand me that black crayon"

```
"bluecar".match(/red|blue/);
```

```
["blue"]
```

```
"blue nice car".match(/red|blue/);
```

```
["blue"]
```

Expresiones Regulares – Sintaxis: {n}

| Sintaxis | Significado |
|----------|--|
| {n} | Donde <i>n</i> es un número positivo. Emparejamiento exactamente <i>n</i> veces. |

Expresión regular: a{2}

Emparejamiento: "Caandy"

Emparejamiento: "Caaandy"

No empareja con: "Candy"

Expresiones Regulares – Sintaxis: {n,m}

| Sintáxis | Significado |
|----------|--|
| {n,m} | Donde <i>n</i> y <i>m</i> son números enteros positivos. Empareja con: al menos <i>n</i> ocurrencias y como máximo <i>m</i> del caracter que presede a la llave {. Cuando <i>n</i> o <i>m</i> valen cero, se pueden omitir. |

Expresión regular: a{1,3}

Emparejamiento: "Candy"

Emparejamiento: "Caaandy"

Emparejamiento: "Caaaandy"

No empareja con: "Cndy"

Expresiones Regulares – Sintaxis: [xyz]

| Sintáxis | Significado |
|----------|---|
| [xyz] | Conjunto de caracteres. Emparejará cualquier caracter del conjunto. |

Expresión regular: [a-d]

Emparejamiento: "candy"

Emparejamiento: "brisket"

Expresión regular: [0-5]

Emparejamiento: "0123456789"

Emparejamiento: "543210"

```
/[a-d]/.test("candy")
```

```
true
```

```
/[0-5]/.test("543210")
```

```
true
```

```
"candy".match(/[a-d]/g)
```

```
["c", "a", "d"]
```

```
"543210".match(/[0-5]/g)
```

```
["5", "4", "3", "2", "1", "0"]
```

Mismas comprobaciones de las 2 expresiones regulares

Expresiones Regulares – Caracteres especiales

Comienzan con el caracter (slash bar) → \

\b: espacio en blanco

\t: tabulador

\d: dígito

\w: caracter

\S: no es un espacio en blanco

\D: no es un dígito

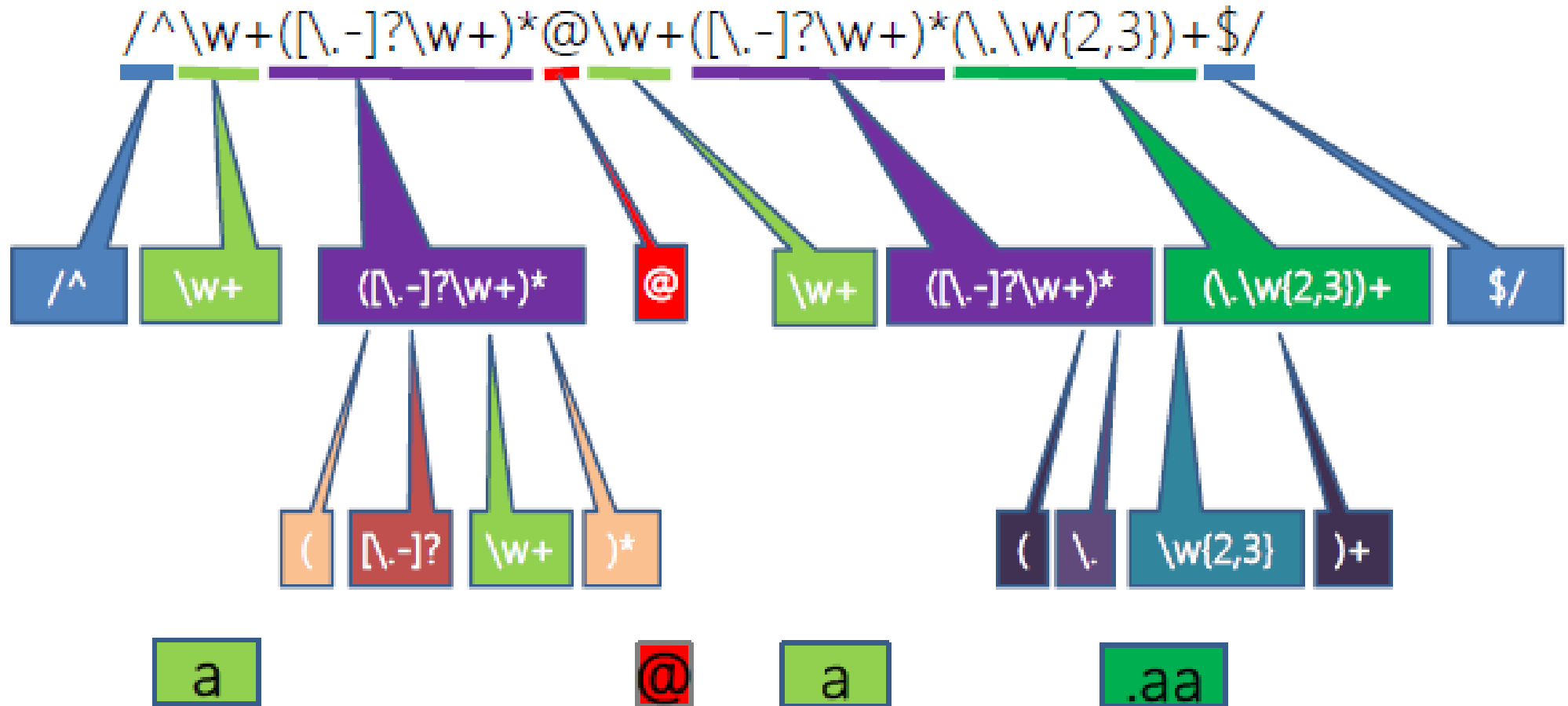
\W: no es un caracter

Cualquier caracter precedido de \ significa que vamos a “escapar” al metacaracter, es decir, queremos que ese metacaracter se interprete como un caracter

Un metacaracter es un caracter que tiene un significado especial o comodines

\? → indica que se quiere interpretar el caracter “?”

Expresiones Regulares – Validar Email



¿Cuál es la dirección email más corta que se permite?

Validar Formulario - Campo de texto obligatorio

Comprobamos que el campo sea cumplimentado y que no haya espacios en blanco.

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

Validar Formulario - Campo de texto numérico

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


Validar Formulario - Campo de selección

```
indice = document.getElementById("opciones").selectedIndex;  
if( indice == null || indice == 0 ) {  
    return false;  
}
```

Validar Formulario - Dirección email

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

Validar Formulario - Fecha

```
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;
}
```

Validar Formulario - Teléfono

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

Validar Formulario - DNI

```
valor = document.getElementById("campo").value;  
if( !(/^(^[0-9]{7,8}\-[A-Z]))$/ .test(valor)) {  
    return false;  
}
```

Validar Formulario - Checkbox

```
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;  
        }  
    }  
}
```

Validar Formulario - 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;
```

```
}
```