



JavaScript



ANEXO 7.-

JQuery:

Validación de formularios



DISTRIBUIDO POR:

CENTRO DE INFORMÁTICA PROFESIONAL S.L.

C/ URGELL, 100
08011 BARCELONA
TFNO: 93 426 50 87

C/ RAFAELA YBARRA, 10
48014 BILBAO
TFNO: 94 448 31 33

www.cipsa.net

**RESERVADOS TODOS LOS DERECHOS. QUEDA PROHIBIDO TODO TIPO DE
REPRODUCCIÓN TOTAL O PARCIAL DE ESTE MANUAL, SIN PREVIO
CONSENTIMIENTO POR EL ESCRITOR DEL EDITOR**

Validación de formularios

Una de las tareas principales de Javascript es la validación de formularios. Esta consiste básicamente en comprobar que los valores y opciones marcadas en los diferentes elementos que componen un formulario sean válidos.

Un ejemplo clásico de validación consiste en comprobar que una caja de texto en la que el usuario debe introducir su DNI no está vacía en el momento de enviar los datos, en cuyo caso; debe cancelarse el envío de éstos al servidor y mostrar un mensaje de advertencia.

Selectores de elementos de formulario

jQuery ofrece pseudoselectores específicos para obtener referencia a los diferentes controles de los formularios:

- **:button** → Selecciona elementos `<button>` y con el atributo `type='button'`
- **:checkbox** → Selecciona elementos `<input>` con el atributo `type='checkbox'`
- **:checked** → Selecciona elementos `<input>` del tipo checkbox seleccionados
- **:disabled** → Selecciona elementos del formulario que están deshabilitados
- **:enabled** → Selecciona elementos del formulario que están habilitados
- **:file** → Selecciona elementos `<input>` con el atributo `type='file'`
- **:image** → Selecciona elementos `<input>` con el atributo `type='image'`
- **:input** → Selecciona elementos `<input>`, `<textarea>` y `<select>`
- **:password** → Selecciona elementos `<input>` con el atributo `type='password'`
- **:radio** → Selecciona elementos `<input>` con el atributo `type='radio'`
- **:reset** → Selecciona elementos `<input>` con el atributo `type='reset'`
- **:selected** → Selecciona elementos `<options>` que están seleccionados
- **:submit** → Selecciona elementos `<input>` con el atributo `type='submit'`
- **:text** → Selecciona elementos `<input>` con el atributo `type='text'`

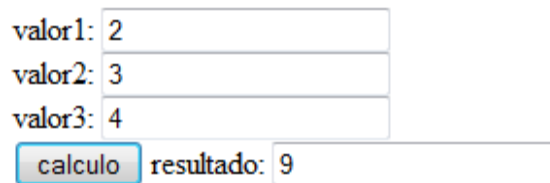
Eventos típicos de elementos de formulario

- **click**: evento que se produce cuando se pincha con el ratón sobre un elemento. Normalmente se utiliza con cualquiera de los tipos de botones que permite definir HTML: `<input type="button">`, `<input type="submit">` y `<input type="image">`.
- **change**: evento asociado al cambio del valor de un elemento de texto: `<input type="text">` o `<textarea>`, o la selección de una opción en una lista desplegable `<select>`. En las listas el evento se dispara en el momento de la selección de una opción. En las cajas de texto se dispara al perder el foco si su valor ha cambiado.

CURSO DE PROGRAMACION JAVASCRIPT

- **focus**: evento que se produce cuando el usuario selecciona un elemento del formulario. Se dice entonces que el elemento tiene el “foco”.
- **blur**: evento complementario de **onfocus** que se produce cuando el elemento pierde el “foco” al seleccionar el usuario otro elemento.

Ejemplo: La siguiente página muestra un formulario provisto de tres cajas de texto y un botón que al ser pulsado muestra el sumatorio de los valores indicados en una cuarta caja de texto de solo lectura:



```
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="Scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      // Seleccion cajas de texto con valores de entrada
      var $cajas = $("input:text:not(#resultado)");
      // Seleccion caja de texto para valor resultado
      var $resultado = $("#resultado");
      // Pulsacion botón de cálculo
      $("input:button").click(function () {
        var sumatorio = 0;
        // Obtencion de valores y calculo de sumatorios
        $cajas.each(function () {
          sumatorio += parseInt($(this).val());
          $resultado.val(sumatorio);
        })
      });
    });
  </script>
</head>
<body>
  <form>
    valor1: <input type="text"><br>
    valor2: <input type="text"><br>
    valor3: <input type="text"><br>
    <input type="button" value="calculo">
    resultado: <input type="text" id="resultado" readonly>
  </form>
</body>
</html>
```

La función de cálculo del sumatorio se ejecuta al producirse el evento “click” sobre el botón con valor “cálculo”. La función recorre las cajas de texto donde se introducen los valores sumando sus valores en la variable “sumatorio” que es finalmente asignada como valor a la caja de texto con *id*=“resultado”.

Ejemplo 2: Se modifica la página anterior de modo que la caja de texto que posea el foco se muestre con el fondo azulado. Para ello se registran los eventos *focus* y *blur* de todas ellas exceptuando la de resultado:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="Scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      var $cajas = $("input:text:not(#resultado)");
      var $resultado = $("#resultado");
      $cajas.focus(function () {
        $(this).css("background-color", "#00ffff");
      })
      $cajas.blur(function () {
        $(this).css("background-color", "#ffffff");
      })
      $("input:button").click(function () {
        var sumatorio = 0;
        $cajas.each(function () {
          sumatorio += parseInt($(this).val());
          $resultado.val(sumatorio);
        })
      })
    });
  </script>
</head>
<body>
  <form>
    valor1: <input type="text"><br>
    valor2: <input type="text"><br>
    valor3: <input type="text"><br>
    <input type="button" value="calcula">
    resultado: <input type="text" id="resultado" readonly>
  </form>
</body>
</html>
```

Manejo de elementos de formulario

Cajas de texto

Los elementos que representan cajas de texto poseen la propiedad **value** que permite tanto obtener el valor introducido por el usuario como modificarlo desde javascript. Empleando JQuery puede obtener y modificarse el valor de una caja de texto mediante la funcion **val()**.

Ejemplo 3: Se modifica la página anterior incluyendo una funcion que comprueba que se hayan introducido valores numéricos correctos en las cajas de texto al calcular el sumatorio. En caso de que alguna tenga un valor nulo o no numérico se tiñe el fondo de rojo y se indica un mensaje de error:

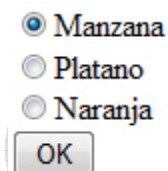
```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="Scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      var $cajas = $("input:text:not(#resultado)");
      var $resultado = $("#resultado");
      // Funcion de validacion de valores introducidos
      function check() {
        var ok = true;
        // Para cada caja de texto
        $cajas.each(function () {
          // Comprobacion de error
          if (!$.isNumeric($(this).val())) {
            $(this).css("background-color", "#ff0000");
            ok = false;
          }
        });
        return ok;
      }
      // Evento de pulsacion de botón de ejecución de cálculo
      $("input:button").click(function () {
        // Comprobacion de valores
        if (check()) {
          // Comprobacion correcta
          var sumatorio = 0;
          $cajas.each(function () {
            sumatorio += parseInt($(this).val());
            $resultado.val(sumatorio);
          })
        } else {
          // Comprobacion no superada
          alert("Se ha encontrado un valor erroneo");
        }
      })
    });
  </script>
</head>
<body>
  <form>
    valor1: <input type="text"><br>
    valor2: <input type="text"><br>
    valor3: <input type="text"><br>
    <input type="button" value="calcula">
    resultado: <input type="text" id="resultado" readonly>
  </form>
</body>
</html>
```

Botones de opción

Los botones de opción permiten seleccionar una opción entre varias de modo que una y sólo una puede ser activa al mismo tiempo. Para ello se asigna el mismo valor al atributo **name** de todos los elementos. Cada objeto correspondiente a un botón de opción en javascript dispone de una propiedad **checked** que indica si está marcado y una propiedad **value** que devuelve el valor asignado al atributo **value**.

En JQuery podemos obtener la opción seleccionada mediante el valor del atributo **name** común y el pseudoselector **:checked**.

Ejemplo: El siguiente código muestra un formulario en el que el usuario puede seleccionar una fruta mediante tres botones de opción. Cuando el usuario pulsa el botón "OK" la página indica si se ha marcado alguna fruta y cuál ha sido.



```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="Scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      // Registro del evento de "click" para el botón
      $("input:button").click(function () {
        // Obtencion de la opcion seleccionada
        $opcion = $("input[name='frutas']:checked");
        // Comprobacion de seleccion de opcion
        if ($opcion.length > 0) {
          // Opcion seleccionada
          alert("La opcion seleccionada " + $opcion.val());
        } else {
          // Ninguna opcion seleccionada
          alert("No has seleccionada nada");
        }
      })
    })
  </script>
</head>
<body>
  <form>
    <input type="radio" value="manzana" name="frutas">Manzana<br>
    <input type="radio" value="platano" name="frutas">Platano<br>
    <input type="radio" value="naranja" name="frutas">Naranja<br>
    <input id="boton" type="button" value="OK">
  </form>
</body>
</html>
```

Casillas de verificación

Las casillas de verificación son equivalentes a los botones de opción pero no son mutuamente excluyentes y pueden marcarse unas independientemente de otras. Al margen de esta diferencia poseen las mismas propiedades que los botones de opción para conocer si están marcados (**checked**) y su valor asociado (**value**).

En JQuery podemos comprobar si la casilla de verificación seleccionada está marcada empleando la funcion `$(id).is()` con el pseudoselector `:checked`.

Ejemplo: El siguiente código muestra un formulario con una casilla de verificación. Cuando el usuario pulsa el botón "OK" se muestra un mensaje indicando si fue seleccionada o no.



```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="Scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#boton").click(function () {
        if ($("#visa").is(":checked")) {
          alert("VISA marcado");
        } else {
          alert("VISA no marcado");
        }
      })
    })
  </script>
</head>
<body>
  <form>
    <body>
      <input type="checkbox" id="visa">PAGO VISA<br>
      <input id="boton" type="button" value="OK">
    </body>
  </form>
</body>
</html>
```


Listas de selección

Las listas de selección permiten seleccionar una o varias opciones. Sus objetos se disponen como elementos **<option>** que pueden seleccionarse.

Ejemplo: Sea la siguiente lista:

```
<select id="frutas" size="3">
  <option value="1.00">Manzana</option>
  <option value="1.30">Platano</option>
  <option value="0.95">Naranja</option>
</select>
```

Para obtener todos los elementos contenidos en la lista empleamos el selector:

```
#frutas option
```

Para obtener el elemento seleccionado añadimos el selector **“:selected”**:

```
#frutas option:selected
```

Para comprobar si se ha seleccionado alguna opción podemos comprobar que la propiedad **length** retorna un valor mayor que 0. Del elemento seleccionado podemos obtener el valor mostrado al usuario empleando la función **\$.html()**, o el valor de su propiedad **value** mediante la función **\$.val()**.

Ejemplo: El siguiente código muestra una lista de selección simple con tres posible frutas a elegir. Cuando se pulsa el botón “OK” se indica si se ha seleccionado alguna fruta indicando cuál ha sido y su coste correspondiente.

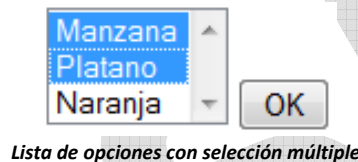
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#boton").click(function () {
        var $seleccion = $("#frutas option:selected");
        // Comprobacion de elemento seleccionado.
        if ($seleccion.length > 0) {
          // Se ha seleccionado un elemento
          alert("El elemento seleccionado es: " + $seleccion.html());
          alert("El valor del producto seleccionado es: " + $seleccion.val());
        } else {
          // No se ha seleccionado ningún elemento.
          alert("No se ha seleccionado ninguna fruta");
        }
      });
    });
  </script>
</head>
<body>
  <select id="frutas" size="3">
    <option value="1.00">Manzana</option>
    <option value="1.30">Platano</option>
    <option value="0.95">Naranja</option>
  </select>
  <input id="boton" type="button" value="OK">
</body>
</html>
```

Lista de selección múltiple

Una lista de selección múltiple añade un atributo **multiple** que permite la selección de múltiples opciones al mismo tiempo. En este caso, el selector para obtener el elemento seleccionado devuelve todas las opciones seleccionadas.

En este caso, la propiedad **length** indica el total de opciones seleccionadas. Puede obtenerse el valor mostrado y el correspondiente al atributo **value** de cada opción seleccionada mediante un bucle implementado a partir de la función **\$.each()**:

Ejemplo: El siguiente código muestra la misma página del ejemplo anterior permitiendo la selección de varias frutas al mismo tiempo. Cuando el usuario pulsa el botón "OK" se muestra un cuadro de alerta con los nombres y precios de las frutas seleccionadas por el usuario:



```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#boton").click(function () {
        var $seleccion = $("#frutas option:selected");
        // Comprobacion de elemento seleccionado.
        if ($seleccion.length > 0) {
          alert("Se han seleccionado: " + $seleccion.length + " opciones.");
          // Bucle de recorrido de opciones seleccionadas
          $seleccion.each(function () {
            // Se ha seleccionado un elemento
            alert("El elemento seleccionado es: " + $(this).html());
            alert("El valor del producto seleccionado es: " + $(this).val());
          })
        } else {
          // No se ha seleccionado ningún elemento.
          alert("No se ha seleccionado ninguna fruta");
        }
      });
    });
  </script>
</head>
<body>
  <select id="frutas" size="3" multiple>
    <option value="1.00">Manzana</option>
    <option value="1.30">Platano</option>
    <option value="0.95">Naranja</option>
  </select>
  <input id="boton" type="button" value="OK">
</body>
</html>
```

Agregar opciones a una lista

jQuery permite eliminar y añadir elementos a una lista. Esto puede resultar útil en aquellos casos en los que las opciones a mostrar dependen de una selección previa.

Ejemplo: Supóngase una web en la que el usuario puede seleccionar un día de la semana en su correspondiente idioma. Para ello la web consta de una lista con 4 idiomas (castellano, inglés, euskera, y francés), y una segunda lista donde deben mostrarse los días de la semana en el idioma seleccionado. La idea es emplear jQuery para detectar la selección de un idioma en la lista de idioma, y agregar los días de la semana en el idioma seleccionada a la segunda lista eliminando los existentes.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
</script>

// Matriz bidimensional de dias por idioma.
var dias = [
  ['lunes', 'martes', 'miercoles', 'jueves', 'viernes', 'sabado', 'domingo'],
  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'satuday', 'sunday'],
  ['astelehena', 'asteartea', 'asteazkena', 'osteguna', 'ostirala', 'larunbata', 'igandea'],
  ['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi', 'samedi', 'dimanche']
];

$(document).ready(function() {
  // Evento de seleccion de idioma en lista idiomas
  $('#idiomas').change(function() {
    // Muestra el valor de la opcion seleccionada
    console.log($(this).val());
    // Muestra el texto de la opcion seleccionada
    console.log($('#idiomas :selected').text());
    // Obtencion del valor del idioma y refresco de la lista de dias
    refrescarListaDias( $(this).val());
  });
});

// Funcion que refresca la lista de dias según el indice del idioma seleccionado
function refrescarListaDias( idioma ) {
  // Obtencion de la lista
  var lista = $('#dias');
  // Elimina todas las opciones actuales.
  $('option', lista).remove();
  // Obtencion matriz de opciones de la lista
  var opciones = lista.prop('options');
  // Recorrido de la submatriz de dias en el idioma correspondiente
  $.each(dias[idioma], function(indx, val) {
    // Agregado de cada dia a la matriz de opciones de la lista
    opciones[opciones.length] = new Option(val, indx);
  });
}

</script>
<title>Insert title here</title>
</head>
<body>
  <div>
    <form>
      Idioma: <select id='idiomas'>
        <option value='0'>Castellano</option>
        <option value='1'>Ingles</option>
        <option value='2'>Euskera</option>
        <option value='3'>Frances</option></select>
      <br/>
      Dias: <select id='dias'></select>
    </form>
  </div>
</body>
</html>
```

Niveles de validación de formularios

La validación de los valores de un formulario puede realizarse bien en el momento en que el usuario rellena los campos o al final al pulsar el botón de envío.

Validación a nivel del control

Si se valida a nivel control y el valor introducido no es válido puede mostrarse un cuadro de diálogo con un mensaje de error y cancelar el cambio de foco para obligar al usuario a corregir el error.

Ejemplo: Supóngase el formulario del ejemplo anterior en el que además de mostrarse el error de validación de cada control al perder el foco, se retiene el mismo en el control hasta introducir un valor considerado válido:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <style>
    .error {
      color: red;
      display: none;
    }
  </style>
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#nombre").blur(function () {
        if (this.validity.valueMissing) {
          $("#err_nombre").html("* Campo obligatorio.").show();
          this.focus(); // Retiene el foco en el control hasta valor válido
        } else {
          $("#err_nombre").html("").hide();
        }
      });
      $("#edad").blur(function () {
        if (this.validity.valueMissing) {
          $("#err_edad").html("* Campo obligatorio").show();
          this.focus(); // Retiene el foco en el control hasta valor válido
        } else if (this.validity.rangeUnderflow) {
          $("#err_edad").html("* Valor muy bajo").show();
          this.focus(); // Retiene el foco en el control hasta valor válido
        } else if (this.validity.rangeOverflow) {
          $("#err_edad").html("* Valor muy alto").show();
          this.focus(); // Retiene el foco en el control hasta valor válido
        } else {
          $("#err_edad").html("").hide();
        }
      });
    });
  </script>
</head>
<body>
  <form name="miformulario" id="miformulario" method="get">
    NOMBRE <input type="text" id="nombre" name="nombre" required>
    <output class="error" id="err_nombre"></output><br />
    EDAD <input type="number" id="edad" name="edad" min="18" max="65" required>
    <output class="error" id="err_edad"></output><br />
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

Validación a nivel de formulario

En la validación a nivel de formulario se utiliza una función de validación general que comprueba los valores introducidos en todos los campos mediante funciones de validación auxiliares y retorna un valor lógico indicando si todos son correctos. Esta función se ejecuta cuando el usuario pulsa el botón de envío y tiene la siguiente estructura:

```
function valida() {
    if ( comprobación_control_1 ) {
        alert("[ERROR control 1]");
        return false; // Fallo
    } else if ( comprobación_error_2 ) {
        alert("[ERROR control 2]");
        return false; // Fallo
    } else if ( comprobación_control_3 ) {
        alert("[ERROR control 3]");
        return false; // Fallo
    } else {
        return true; // Todo OK
    }
}
```

Ejemplo: El siguiente código muestra la validación a nivel de formulario de los campos del formulario del ejemplo anterior:

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="utf-8" />
    <script src="scripts/jquery-3.1.0.js"></script>
    <script>
        $(function () {
            function valida_novacio(valor) { // Función de validación valor vacío
                return valor.length > 0;
            }
            function valida_numerico(valor) { // Función de validación numérico
                return $.isNumeric(valor);
            }
            $("#formulario").submit(function (ev) {
                if (!valida_novacio($("#nombre").val())) {
                    alert("El campo nombre es obligatorio");
                    ev.preventDefault(); // Cancela envío de datos
                }
                if (!valida_numerico($("#edad").val())) {
                    alert("El campo edad no es un número");
                    ev.preventDefault(); // Cancela envío de datos
                }
            });
        });
    </script>
</head>
<body>
    <form id="formulario" method="post">
        nombre: <input type="text" id="nombre"><br />
        edad: <input type="text" id="edad">
        <input type="submit" id="btn_envio" value="ENVIAR">
    </form>
</body>
</html>
```

Cuando se lanza el evento de envío del formulario (**submit**), se invoca a las funciones auxiliares **valida_novacio()** y **valida_numero()** pasando como argumento los valores introducidos por el usuario en las cajas de texto a comprobar. Cada función recibe como argumento el valor introducido por el usuario y devuelve un valor lógico indicando si es o no correcto. Si alguna de estas funciones retorna un valor lógico *falso* se muestra un mensaje de error y se cancela el evento de envío de formulario.

Funciones de validación como ejemplo:

Validación de texto obligatorio → Retorna un valor lógico 'cierto' si el valor es dado está vacío o es nulo.

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

Validación de valor numérico → Retorna un valor lógico 'cierto' si el valor dado es un valor numérico válido.

```
function esNumero( valor ) {
    return !isNaN(valor);
}
```

Validación de dirección de correo electrónico → Retorna un valor lógico 'cierto' si el argumento dado es una dirección de e-mail correctamente formateada.

```
function esMail( valor ) {
    return (/w+([-.']w+)*@w+([-.']w+)*\.w+([-.']w+)/.test(valor));
}
```

(*) Esta función emplea una expresión regular que comprueba el formato de una dirección de correo electrónico pero no si la cuenta realmente existe.

Validación de número de DNI → Retorna un valor lógico 'cierto' si el argumento dado es un DNI correctamente formateado. Se comprueba el formato y la letra:

```
function esDNI( valor ) {
    var ok = true;
    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', 'I'];
    if( !(/^\d{8}[A-Z]$/ .test(valor)) ) {
        ok = false;
    } else if(valor.charAt(8) != letras[(valor.substring(0, 8))%23]) {
        ok = false;
    }
    return ok;
}
```

Validación de número de teléfono → Retorna un valor lógico 'cierto' si el argumento indicado es un número de teléfono compuesto por 9 cifras numéricas.

```
function esTelefono( valor ) {
    return (/^\d{9}$/ .test(valor));
}
```

Validación de formularios en HTML5

HTML5 introduce múltiples mecanismos de validación automática que no requieren ya del uso de Javascript:

- Validación de datos por el tipo asociado (*type="mail/url/number..."*)
- Validación de campos requeridos (atributo *required*)
- Validación de valores con un determinado formato (atributo *pattern*).

Sin embargo, cuando se tratan de validaciones más complejas como comprobar el valor de varios campos, o validar el resultado de un cálculo a partir de los valores introducidos por el usuario; entonces es necesario emplear javascript. Para estos casos HTML5 define una serie métodos y eventos que permiten tanto ejecutar una función de javascript cuando la validación automática falla, como comprobar en cualquier momento la validez del valor introducido en un elemento.

Definición de error personalizado

HTML5 automatiza la validación de ciertos controles de formulario en función de su tipo y el uso de ciertos atributos. De modo que si algún valor no es válido se cancela el envío mostrando un mensaje de error.

Para definir un tipo de validación adicional o más específica de las realizadas automáticamente por HTML5 debe emplearse *Javascript*. Esto se hace comprobando el valor del elemento al cambiar éste y si no es correcto asignándole un mensaje de error mediante el método ***setCustomValidity()***. Este mensaje se mostrará al intentar enviar los datos al formulario. Si se invoca al método indicando como argumento una cadena vacía se entiende que su valor es válido y el formulario se enviará.

Ejemplo: Supóngase un formulario con un campo de texto en el que el usuario debe introducir un valor de 8 caracteres. En caso contrario debe mostrarse el mensaje de error *"Introduzca valor de 8 caracteres"* al enviar el formulario.

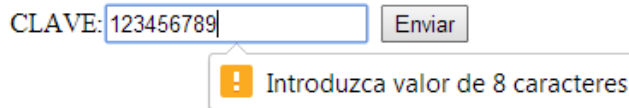
```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#formulario input[name='clave']").change(function () {
        if ($(this).val().length > 8) {
          this.setCustomValidity("Introduzca un valor de 8 caracteres");
        } else {
          this.setCustomValidity("");
        }
      });
    });
  </script>
</head>
```

```
<body>
  <form id="formulario">
    CLAVE:<input type="text" name="clave">
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

Cada vez que el usuario introduce un valor (evento **change**) en la caja de texto se comprueba la longitud del valor insertado y si es diferente de 8 le asigna el mensaje de error *"Introduzca valor de 8 caracteres"*.

```
campo.setCustomValidity("Introduzca valor de 8 caracteres");
```

Hecho esto, cuando el usuario pulse el botón de envío los datos no se enviarán y se mostrará el mensaje de error indicado:



Si por el contrario el valor se asigna una cadena vacía al método **setCustomValidity()** para indicar que el control no tiene error.

```
campo.setCustomValidity("");
```

Añadir código Javascript en caso de fallo

Puede añadirse una función de javascript para que se ejecute en caso de fallo de la validación automática capturando el evento **invalid** del formulario. Este evento se dispara cuando el usuario pulsa el botón de envío y la validación de algún elemento es incorrecta.

Ejemplo: Supóngase un formulario en el que el usuario debe introducir obligatoriamente su nombre y edad, y donde la edad debe ser además un valor numérico válido:

NOMBRE

EDAD

Enviar

Se pide que cuando se pulse el botón de envío, si algún campo no es válido, se muestre un cuadro de diálogo indicando el nombre campo incorrecto y se tiña de rojo su color de fondo como indicación para el usuario.


```

<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#formulario input:not(:submit)").on("invalid", function () {
        $(this).css("background-color", "#ff0000");
      });
    })
  </script>
</head>
<body>
  <form name="formulario" id="formulario" method="get">
    NOMBRE <input type="text" id="nombre" name="nombre" required><br />
    EDAD <input type="number" id="edad" name="edad" required><br />
    <input type="submit" value="Enviar">
  </form>
</body>
</html>

```

En este código se ha capturado el evento **invalid** del formulario de modo que cuando se pulsa el botón de envío y falla se asigna un estilo de color de fondo rojo al elemento incorrecto. Es importante tener presente que el código Javascript no sustituye a la validación automática del formulario, por lo que los mensajes de error asociados a cada elemento se mostrarán igualmente:

NOMBRE

EDAD

Enviar

Completa este campo

Visualización del error

Obtención del estado de validez de un elemento

HTML5 permite mediante Javascript conocer el estado de validez de un elemento. Para ello se dispone del objeto **ValidityState** retornado por la propiedad **validity** de cualquier elemento. Este objeto posee las siguientes propiedades que devuelven un valor lógico *cierto/falso*:

- **valid** → Devuelve cierto si el valor del campo es válido. En caso contrario devuelve falso.
- **valueMissing** → Devuelve cierto si el campo posee el atributo *required* y no se ha introducido ningún valor.
- **patternMismatch** → Devuelve cierto si el elemento posee el atributo *pattern* y el valor presente no cumple con la expresión regular especificada.
- **tooLong** → Devuelve cierto si el valor introducido en el campo es más largo que el valor especificado en el atributo *maxlength*.
- **rangeUnderflow** → Devuelve cierto si el valor introducido en el campo de tipo numérico (*type="number"*) es inferior al especificado en el atributo *min*.
- **rangeOverflow** → Devuelve cierto si el valor introducido en el campo de tipo numérico (*type="number"*) es superior al especificado en el atributo *max*.
- **stepMismatch** → Devuelve cierto si el valor introducido en el campo de tipo numérico (*type="number"*) no se corresponde con los valores de los atributos *max*, *min*, *step*.
- **customError** → Devuelve cierto si se ha asignado al elemento un mensaje de error mediante el método *setCustomValidity()*.

Estas propiedades pueden invocarse en cualquier sin esperar a que el usuario pulse el botón de envío, y resultan útiles para indicar al usuario los errores mientras va rellenando los campos de un formulario en vez de hacerlo al final.

CURSO DE PROGRAMACION JAVASCRIPT

Ejemplo: Supóngase que tenemos un formulario donde el usuario debe introducir obligatoriamente su nombre y edad, y la edad debe ser además un valor numérico comprendido entre 18 y 65:

NOMBRE * Campo obligatorio

EDAD * Valor muy bajo

Se desea que cuando va cambiando de foco entre los campos se validen sus valores y se muestren notificaciones de error al margen si es necesario:

```
<!DOCTYPE html>
<html>
<head>
  <title></title>
  <meta charset="utf-8" />
  <style>
    .error {
      color: red;
      display: none;
    }
  </style>
  <script src="scripts/jquery-3.1.0.js"></script>
  <script>
    $(function () {
      $("#nombre").blur(function () {
        if (this.validity.valueMissing) {
          $("#err_nombre").html("* Campo obligatorio.").show();
        } else {
          $("#err_nombre").html("").hide();
        }
      });
      $("#edad").blur(function () {
        if (this.validity.valueMissing) {
          $("#err_edad").html("* Campo obligatorio").show();
        } else if (this.validity.rangeUnderflow) {
          $("#err_edad").html("* Valor muy bajo").show();
        } else if (this.validity.rangeOverflow) {
          $("#err_edad").html("* Valor muy alto").show();
        } else {
          $("#err_edad").html("").hide();
        }
      });
    });
  </script>
</head>
<body>
  <form name="miformulario" id="miformulario" method="get">
    NOMBRE <input type="text" id="nombre" name="nombre" required>
    <output class="error" id="err_nombre"></output><br />
    EDAD <input type="number" id="edad" name="edad" min="18" max="65" required>
    <output class="error" id="err_edad"></output><br />
    <input type="submit" value="Enviar">
  </form>
</body>
</html>
```

Cuando cada una de las cajas de texto pierden el foco (evento **blur**), se comprueban las propiedades de su estado de validación a través de la propiedad **validity**, y se muestra un mensaje de error mediante los campos **<output>** correlativos.

Ejercicio

Crea una página web provista de un formulario para realizar compras online para una frutería. El formulario debe contener los siguientes controles:

- 1 caja de texto para el nombre del comprador.
- 1 lista con las siguientes opciones: “Naranjas”, “Manzanas”, “Limones”, “Tomates”.
- 2 botones de opción *<option>*: “VISA” y “MASTERCARD”.
- 1 caja de texto para el peso en kilos.
- 1 botón con el texto “Calcular”

El usuario debe introducir obligatoriamente un nombre, seleccionar una fruta, una opción de compra e indicar el peso mediante un valor numérico válido entre 0 y 10Kg.

Cuando el usuario pulse el botón de envío debe calcularse y mostrarse el importe total de la compra según los siguientes precios estipulados:

- Naranjas: 3€ / kg
- Limones: 2.25€ / kg
- Manzanas: 2.50€/kg
- Tomates: 4.25€/kg

Si el usuario compra más de 5 Kg se le aplica un descuento por volumen de compra del 4% si paga con VISA, y del 2.5% si es MASTERCARD.

Consejo: Para hacer este ejercicio es recomendable crear las siguientes funciones:

- *ValidarVacio()* → retorna cierto si la caja de texto del nombre tiene un valor. En caso contrario muestra el mensaje “No se ha indicado el nombre”, y retorna un valor falso.
- *ValidarValor()* → retorna cierto si la caja de texto del peso tiene un valor numérico comprendido entre 0.0 y 10.0. En caso contrario muestra el mensaje “Peso incorrecto.”, y retorna un valor falso.
- *ValidarSeleccion()* → retorna cierto si se ha seleccionado una fruta. En caso contrario muestra el mensaje “Debe seleccionar una fruta” y retorna falso.
- *Calcular()* → Función que es llamada al pulsarse el botón Calcular. Esta función comprueba llama primero a las funciones de validación: ValidarVacio, ValidarValor y ValidarSeleccion. Si todas las funciones devuelven un valor lógico cierto entonces se calcula el precio de la compra conforme a la fruta, el peso y la opción de pago seleccionados, mostrando el resultado por pantalla.