



UD3 HTML Y CSS.

# DISEÑO DE INTERFACES WEB



## 3.4 Formularios HTML5. Incrustación de fuentes y otras

# Contenidos

1. Objetivos
2. Introducción
3. HTML5. Formularios
4. CSS3
5. Ejercicios

# 1. Objetivos

En este cuarto tema se pretenden conseguir los siguientes objetivos:

- Crear **formularios** utilizando los nuevos tipos de entrada para formularios HTML5.
- Utilizar los nuevos atributos de HTML5 para **validación de formularios**. Validar formularios utilizando la nueva API Forms de HTML5.
- Aplicar **estilos** a los formularios con las nuevas pseudo-clases de CSS3. **Incrustar fuentes** en nuestras páginas web.
- Diseñar webs utilizando **múltiples fondos de imagen** para un mismo elemento.

## 2. Introducción

En este tema hablaremos, principalmente, sobre el **uso de formularios**.

Veremos los **nuevos tipos de entrada** que se han introducido en HTML5 (search, email, url, tel, etc.), que nos ayudarán a indicar que tipo de datos se espera que el usuario introduzca en los campos del formulario.

Veremos también los **nuevos atributos** que se han creado para ayudar en la **validación** de estos campos (required, pattern, etc.), o para **ampliar las funcionalidades** de los mismos (placeholder, multiple, autofocus, etc.).

También veremos una nueva API JavaScript que incorpora HTML5 (**API Forms**), que nos permitirá realizar validaciones más complejas de forma sencilla.

## 2. Introducción

En lo que se refiere a la parte de CSS3, hablaremos sobre las **nuevas pseudo-clases** que se introducen en el estándar para aplicar estilos a los elementos de los formularios según sean requeridos, con datos válidos o con datos inválidos (**:required**, **:valid** e **:invalid**).

También veremos cómo mejorar nuestros diseños utilizando **varias imágenes de fondo** en un mismo elemento y cómo **incrustar fuentes** en nuestros sitios web para poder utilizarlas sin preocuparnos de que el usuario las tenga instaladas en su equipo.

### 3. HTML5. Formularios

La Web 2.0 está completamente **enfocada al usuario**, y por tanto **relacionado con interfaces**: cómo hacerlas más intuitivas, más prácticas y, por supuesto, más atractivas. Los **formularios son las interfaces más importantes**, permiten a los usuarios insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación.

HTML5 ha introducido **nuevos elementos** de formulario: tipos de `<input>`, atributos y otras características. La diferencia es que, mientras que antes teníamos que recurrir a JavaScript para crear estos **comportamientos**, ahora muchos de ellos están disponibles directamente en el navegador; todo lo que tenemos que hacer es establecer un atributo en el elemento adecuado para que estén disponibles.

# Nuevos atributos para formularios

La estructura del formulario y sus atributos son igual que en especificaciones previas pero con **nuevos atributos para <form>**:

- **autocomplete**: Puede tomar dos valores: **on** (valor por defecto) y **off**. Puede ser utilizado en el elemento <form> o en cualquier <input> independientemente. Es una buena idea **desactivar esta funcionalidad en campos sensibles**, como tarjetas de crédito o números de cuenta, y para aquellos campos cuya información nunca va a ser reutilizada, como CAPTCHA.
- **novalidate**: En HTML5, los formularios son **automáticamente validados**. Para evitar este comportamiento, podemos usar el atributo **novalidate**. Para lograr lo mismo para elementos <input> específicos, existe otro atributo llamado **formnovalidate**. Ambos atributos son booleanos, **ningún valor tiene que ser especificado** (su presencia es suficiente para activar su función). El valor del atributo **formnovalidate** sobrescribe el del atributo **novalidate** del formulario.



## Nuevos tipos de <input>

Los elementos más importantes en los formularios son los <input>. Estos elementos pueden cambiar sus características gracias al **atributo type** (tipo). Este atributo determina **qué clase de entrada se espera** que introduzca el usuario. Los tipos disponibles hasta el momento eran: text (para textos en general) y sólo unos pocos más específicos como password o submit. HTML5 ha añadido nuevos tipos, incrementando de este modo las posibilidades de entrada.

En HTML5 estos nuevos tipos especifican qué clase de entrada se espera y le dicen al navegador qué debe hacer con la información recibida. El navegador **procesará** los datos introducidos de acuerdo al valor del atributo type y **validará** la entrada o no. Veamos algún ejemplo:

# Nuevos tipos de <input>

## Tipo search

El tipo search (búsqueda) no controla la entrada, es sólo una **indicación para los navegadores**. Al detectar este tipo de campo algunos navegadores cambiarán el diseño del elemento para hacerle ver al usuario cuál es el propósito del campo. Otros, como Chrome, añaden la posibilidad de vaciar el campo de búsqueda haciendo click en un aspa que aparece a la derecha del campo.

```
<form id="formbusqueda" method="get">  
  <input type="search" id="busqueda" name="busqueda">  
  <input type="submit" value="buscar">  
</form>
```

En Chrome, sin aplicar ningún estilo, se mostraría:



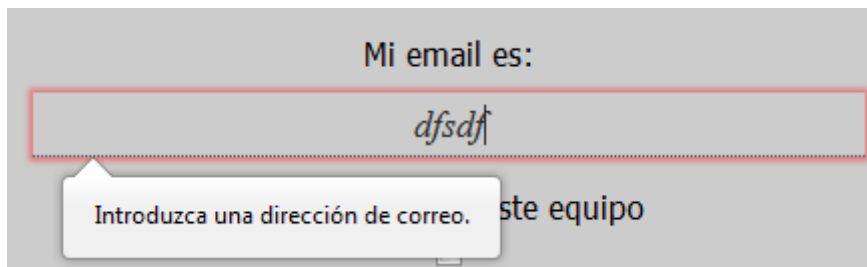
# Nuevos tipos de <input>

## Tipo email

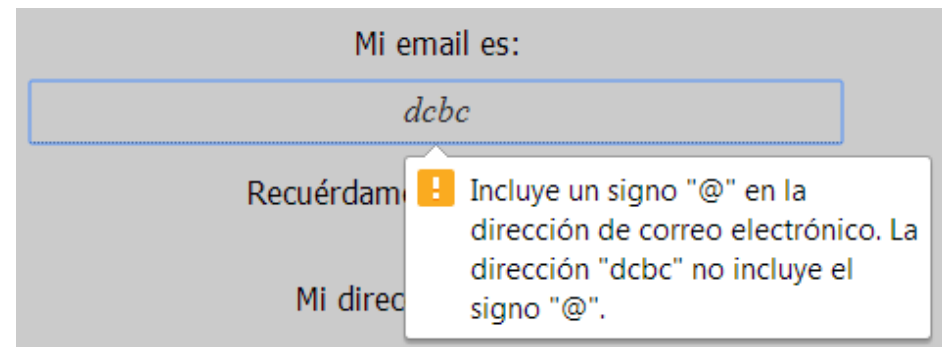
Hasta HTML5 sólo disponíamos del tipo text, por lo que teníamos que controlar con JavaScript que el texto ingresado correspondía a un email válido. Ahora **el navegador se hace cargo** de esto con el nuevo tipo email.

```
<label for="email">Mi email es:</label>
```

```
<input type="email" id="email" name="email">
```



Mensaje de error en Firefox



Mensaje de error en Chrome

Para cambiar este mensaje de error utilizaremos el método `setCustomValidity("Mensaje")` de JavaScript.

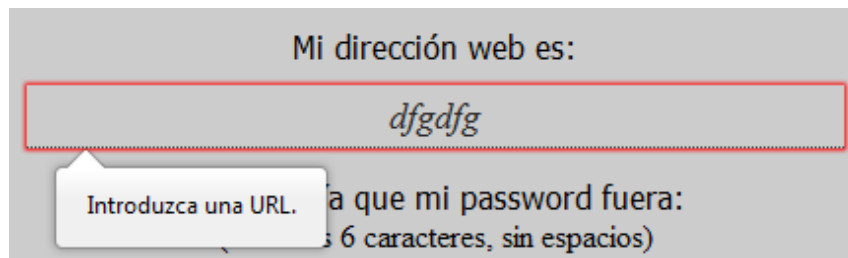
# Nuevos tipos de <input>

## Tipo url

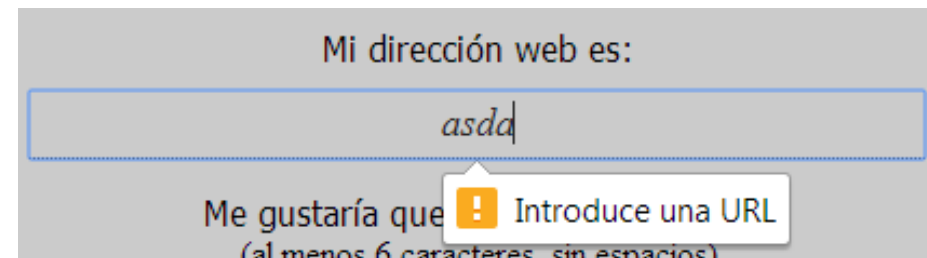
Similar al tipo email, pero es específico para direcciones web. Está destinado a recibir **sólo URLs absolutas** (protocolo, servidor, ruta) y retornará un error si el valor es inválido.

```
<label for="url">Mi dirección web es:</label>
```

```
<input type="url" id="url" name="url">
```



Mensaje de error en Firefox



Mensaje de error en Chrome

Sólo se valida el formato general de la URL, por ejemplo, **q://example.xyz** será válido, aunque q:// no sea un protocolo real. Más adelante, veremos cómo podemos resolver este problema y validar formatos más específicos.

# Nuevos tipos de <input>

## Tipo tel

Este tipo de campo es para **números telefónicos**. A diferencia de los tipos email y url, el tipo tel **no requiere ninguna sintaxis** en particular (cada país tiene un formato determinado para los números de teléfono).

Es sólo una indicación para el navegador en caso de que necesite hacer ajustes de acuerdo al dispositivo en el que la aplicación se ejecuta, por ejemplo, cuando se muestra el teclado en un dispositivo móvil.

# Nuevos tipos de <input>

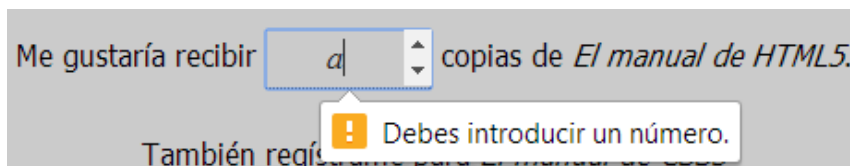
## Tipo number

Se utiliza para pedir entradas numéricas.

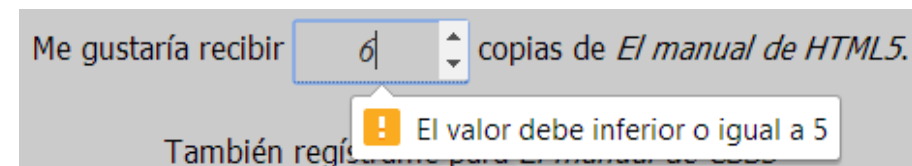
Algunos navegadores muestran este tipo de campo como un “spinner” con una flecha arriba y otra abajo para incrementar o decrementar. Atributos

- min: Mínimo valor aceptado.
- max: Máximo valor aceptado.
- Step: Incremento/decremento en cada paso. Por defecto es 1.

```
<label for="cantidad">Me gustaría recibir <input type="number" name="cantidad" id="cantidad" value="1" min="0" max="5"> copias de <cite>El manual de HTML5</cite>.</label>
```



Error número erróneo en Chrome



Error número fuera de rango en Chrome

# Nuevos tipos de <input>

## Tipo range

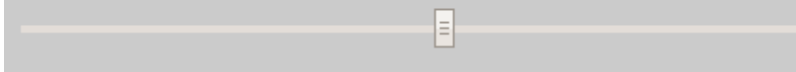
Permite seleccionar un valor a partir de una **serie de valores o rango**. Normalmente, se muestra como una **barra deslizable** (slider). Como en el tipo number, se pueden utilizar los atributos min, max y step. Es ideal para entradas donde esperas un **número no necesariamente preciso**, por ejemplo, en una encuesta de satisfacción para clientes o cosas por el estilo.

El valor por defecto ser el valor medio del rango indicado.

```
<label for="conocimientos">En una escala de 1 a 10, mis conocimientos de HTML5 son:</label>
```

```
<input type="range" min="1" max="10" id="conocimientos" name="conocimientos">
```

En una escala de 1 a 10, mis conocimientos de HTML5 son:



Tipo range en Chrome

En una escala de 1 a 10, mis conocimientos de HTML5 son:



Tipo range en Firefox

# Nuevos tipos de <input>

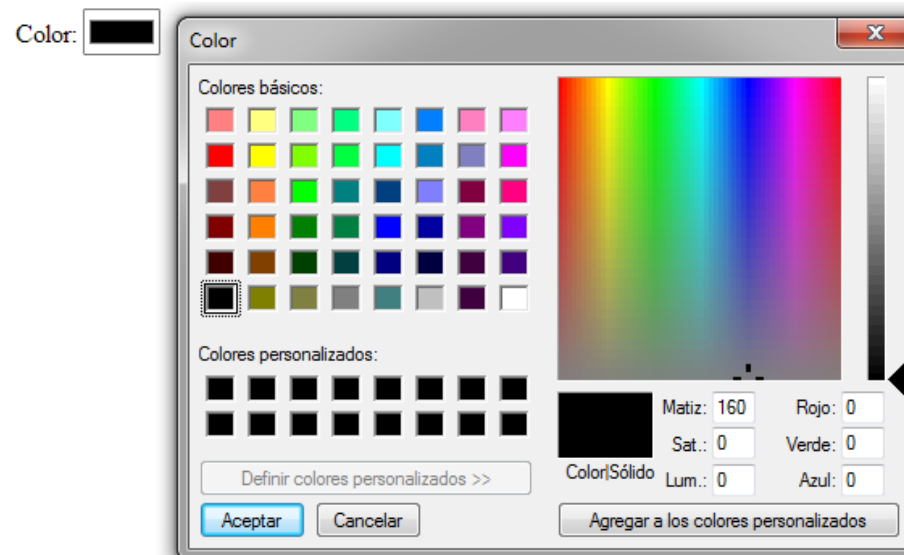
## Tipo color

Permite **introducir colores en formato hexadecimal** (#FEFEFE). Para ello, el navegador nos mostrará una **paleta de colores**, entre los cuales podremos seleccionar el que nos interese.

```
<label for="campoColor">Color: </label>
```

```
<input id="campoColor" name="campoColor" type="color">
```

El ejemplo, después de pulsar el botón que abre la ventana para seleccionar el color, se muestra en Chrome de la siguiente forma:





# Nuevos tipos de <input>

## Tipos de <input> para fechas y hora

Hay varios nuevos tipos de entrada para fecha y hora: date, datetime, datetime-local, month, time, y week. Todas las entradas aceptan datos formateados según la norma ISO 8601.

Los navegadores que soporten estos tipos de datos los mostrarán con un **campo de tipo calendario**, mediante el cual podremos desplegar y seleccionar la fecha con la ayuda del ratón.

- date: comprende la **fecha** (día, mes, año), pero no la hora.

```
<label for="fecha">Fecha: </label>
```

```
<input id="fecha" name="fecha" type="date">
```

Fecha: 13/03/2014 x ▴ ▾ ▼

marzo de 2014 ◿ ◀ ● ▶

| lun | mar | mié | jue | vie | sáb | dom |
|-----|-----|-----|-----|-----|-----|-----|
| 24  | 25  | 26  | 27  | 28  | 1   | 2   |
| 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| 10  | 11  | 12  | 13  | 14  | 15  | 16  |
| 17  | 18  | 19  | 20  | 21  | 22  | 23  |
| 24  | 25  | 26  | 27  | 28  | 29  | 30  |
| 31  | 1   | 2   | 3   | 4   | 5   | 6   |

# Nuevos tipos de <input>

## Tipos de <input> para fechas y hora

- month: Sólo incluye el mes y año. Por ejemplo, febrero de 2014.

<label for="mes">Mes: </label>

<input id="mes" name="mes" type="month">

Mes:

marzo de 2014

| lun | mar | mié | jue | vie | sáb | dom |
|-----|-----|-----|-----|-----|-----|-----|
| 24  | 25  | 26  | 27  | 28  | 1   | 2   |
| 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| 10  | 11  | 12  | 13  | 14  | 15  | 16  |
| 17  | 18  | 19  | 20  | 21  | 22  | 23  |
| 24  | 25  | 26  | 27  | 28  | 29  | 30  |
| 31  | 1   | 2   | 3   | 4   | 5   | 6   |

- week: Indicaremos el número de la semana (del 1 al 52) y el año. Por ejemplo, Semana 11 de 2014.

<label for="semana">Semana: </label>

<input id="semana" name="semana" type="week">

Semana:

marzo de 2014

| Semana | lun | mar | mié | jue | vie | sáb | dom |
|--------|-----|-----|-----|-----|-----|-----|-----|
| 9      | 24  | 25  | 26  | 27  | 28  | 1   | 2   |
| 10     | 3   | 4   | 5   | 6   | 7   | 8   | 9   |
| 11     | 10  | 11  | 12  | 13  | 14  | 15  | 16  |
| 12     | 17  | 18  | 19  | 20  | 21  | 22  | 23  |
| 13     | 24  | 25  | 26  | 27  | 28  | 29  | 30  |
| 14     | 31  | 1   | 2   | 3   | 4   | 5   | 6   |

# Nuevos tipos de <input>

## Tipos de <input> para fechas y hora

- time: Nos permitirá introducir la hora, teniendo en cuenta que la hora irá de 0 a 23. Por ejemplo, 22:05.

<label for="hora">Hora: </label>

Hora:

<input id="hora" name="hora" type="time">

- datetime: Podremos indicar la **fecha y la hora** separadas por una "T" y seguidas, o bien por una "Z" para representar UTC (Coordinated Universal Time), o bien por una zona horaria especificada por un carácter "+" o un carácter "-". Por ejemplo, 17/03/2014T10:45+1:00.

- datetime-local: Es idéntico al datetime, excepto, que se omite la zona horaria. Por ejemplo, 17/03/2014T10:45.

<label for="fechahora">Fecha-hora: </label>

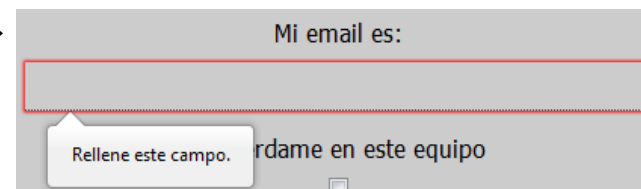
<input id="fechahora" name="fechahora" type="datetime-local">

Fecha-hora:

## Nuevos atributos para los campos del formulario

- **required**: Este atributo booleano no deja que el formulario se envíe si el campo está **vacío**. Por ejemplo, el tipo email anterior:

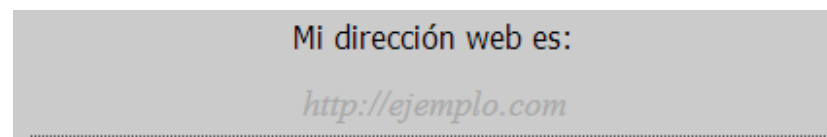
```
<input type="email" id="email" name="email" required>
```



Un formulario con el título "Mi email es:". Debajo hay un campo de texto vacío con un borde rojo. Una etiqueta de error blanca con borde gris dice "Rellene este campo." y apunta al campo. A la derecha del campo, el texto "rdame en este equipo" está parcialmente visible.

- **Placeholder**: representa una **sugerencia** corta, palabra o frase para ayudar al usuario a ingresar la información correcta.

```
<input type="url" id="url" name="url" placeholder="http://ejemplo.com">
```

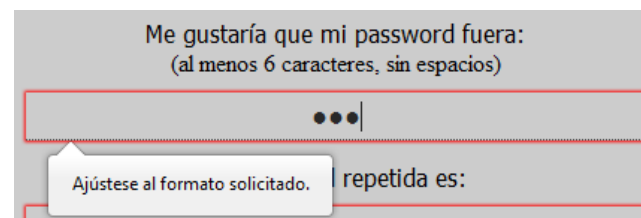


Un formulario con el título "Mi dirección web es:". Debajo hay un campo de texto que contiene el texto "http://ejemplo.com" en un color gris más claro que el fondo, indicando que es un placeholder.

- **pattern**: Se utiliza para propósitos de validación. Usa expresiones regulares para personalizar reglas de validación.

```
<p>(al menos 6 caracteres, sin espacios)</p>
```

```
<input type="password" id="password1" name="password1" required pattern="\S{6,}">
```

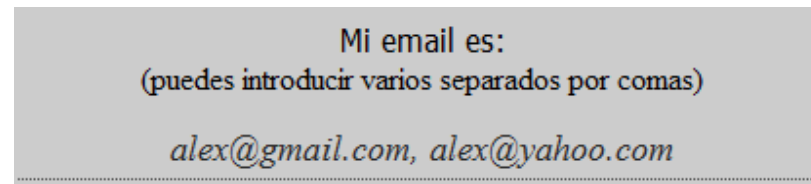


Un formulario con el título "Me gustaría que mi password fuera: (al menos 6 caracteres, sin espacios)". Debajo hay un campo de contraseña con tres puntos "..." y un cursor. Una etiqueta de error blanca con borde gris dice "Ajústese al formato solicitado." y apunta al campo. A la derecha del campo, el texto "repetida es:" está parcialmente visible.

## Nuevos atributos para los campos del formulario

- **multiple**: Atributo booleano que puede ser usado en algunos tipos de campo (por ejemplo, email o file) para **permitir entradas múltiples** en el mismo campo separados por comas.

```
<input type="email" id="email" name="email" required multiple>
```



Mi email es:  
(puedes introducir varios separados por comas)

*alex@gmail.com, alex@yahoo.com*

- **form**: Permite declarar elementos para un **formulario fuera de las etiquetas <form>**, indicando el id del formulario.

```
<input type="text" name="apellidos" form="formulario">
```

- **autofocus**: Indica qué campo (sólo uno) del formulario debe **recibir el foco** en cuanto la página sea cargada.

```
<input type="text" id="nombre" name="nombre" autofocus>
```

- **readonly**: Hace que el usuario **no pueda editar** el campo del formulario.

# Otros tipos de campos para formularios HTML5

## El elemento <datalist> y el atributo list

<datalist>: **lista de ítems** que, con la ayuda del atributo `list`, será usada como **sugerencia** en un campo del formulario.

```
<datalist id="informacion">  
  <option value="11818" label="Inf. 1 11818">  
  <option value="11824" label="Inf. 2 11824">  
</datalist>
```

Este elemento utiliza el elemento <option> en su interior para crear la lista de datos a sugerir. Con la lista ya declarada, lo único que resta es **referenciarla desde un elemento <input>** usando el atributo `list` como se muestra a continuación:

# Otros tipos de campos para formularios HTML5

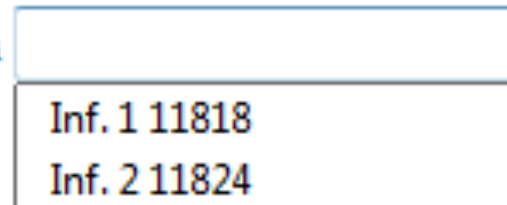
## El elemento <datalist> y el atributo list

```
<label for="telefono">Teléfono información</label>
```

```
<input type="tel" name="telefono" id="telefono" list="informacion">
```

El elemento informacion mostrará **posibles valores** para que el usuario elija.

Teléfono información



|              |
|--------------|
| Inf. 1 11818 |
| Inf. 2 11824 |

Es importante observar que lo que se muestra como sugerencia es el valor del **atributo label** del elemento <option>, pero lo que se carga en el campo al seleccionar la sugerencia es el valor del **atributo value**.

# Otros tipos de campos para formularios HTML5

## Los elementos <progress> y <meter>

<progress> se utiliza para describir el **estado actual de un proceso** que va cambiando hasta llegar a completarse. La barra de **progreso de descarga** es un ejemplo de progreso.

Puede tener un atributo max para indicar el punto en el cual la tarea se completará, y un atributo value para indicar el estado actual de la tarea, aunque ambos atributos son opcionales.

<p>Progreso de la descarga:

```
<progress value="22" max="100"></progress> </p>
```

Progreso de la descarga: 

Normalmente, actualizaremos los valores del elemento <progress> mediante **JavaScript para cambiar dinámicamente** el valor del porcentaje a medida que avanza la tarea.



# Otros tipos de campos para formularios HTML5

## Los elementos <progress> y <meter>

<meter> representa **un elemento cuyo rango se conoce**, lo que significa que tiene definido valores mínimos y máximos. Un ejemplo sería el **espacio disponible de un disco**.

El elemento <meter> tiene siete atributos asociados:

| Atributo | Descripción  |
|----------|--|
| form     | Especifica el id del <form> al que pertenece el elemento <meter> |
| high     | Especifica el intervalo que es considerado como un valor alto.   |
| low      | Especifica el intervalo que es considerado como un valor bajo.   |
| max      | Especifica el valor máximo del intervalo.                        |
| min      | Especifica el valor mínimo del intervalo.                        |
| optimum  | Especifica cuál es el valor óptimo.                              |
| value    | Especifica el valor actual, este atributo es obligatorio.        |

<p>uso de disco actual:

```
<meter value="63" min="0" max="320" low="10" high="300" title="gigabytes">63 GB</meter>
```

</p>

## Otros tipos de campos para formularios HTML5

### El elemento <output>

El propósito del elemento <output> es **visualizar el resultado de un cálculo**.

Se debe utilizar cuando queramos que el usuario pueda **ver el valor, pero no modificarlo** directamente, y cuando el valor se puede derivar de otros valores introducidos en el formulario. Un ejemplo de uso serían los impuestos calculados de un pedido en un carrito de la compra.

El valor del elemento <output> está contenido entre la etiquetas de apertura y de cierre. Por lo general, se utilizará **JavaScript** para actualizar este valor.

## 4. CSS3

### Aplicando estilos a campos requeridos o inválidos


Podemos aplicar estilos a elementos de formularios **requeridos** con la pseudo-clase `:required`. También podemos aplicar estilos a campos válidos **o no válidos** con las pseudo-classes `:valid` e `:invalid`.

Con estas pseudo-classes, podemos proporcionar **indicaciones visuales** a los usuarios que muestren qué campos son obligatorios o si la validación es correcta. Por ejemplo, para que en nuestro formulario de registro aparezca un asterisco (\*) en los campos que sean requeridos, haremos lo siguiente:

```
input { background: transparent no-repeat top right; }  
input:required { background-image: url('../images/required.png'); }
```

Mi email es:  
(puedes introducir varios separados por comas)

.....



## Aplicando estilos a campos requeridos o inválidos

Otras veces, no nos interesará aplicar el **mismo estilo a todos** los tipos de input, Para solucionar esto, usamos la pseudo-clase :not.

```
input:not([type=range]):not([type=submit]):not([type=button]):not([type=checkbox])
```

```
{  
    background: transparent no-repeat top right;  
}
```

```
input:not([type=range]):not([type=submit]):not([type=button]):not([type=checkbox]):required
```

```
{  
    background-image: url('../imgs/required.png');  
}
```

Con estas reglas, evitamos que se inserte la imagen del asterisco, en **aquellos tipos de entrada que no nos interesa**.

# Aplicando estilos a campos requeridos o inválidos

Lo mismo podemos hacer para los elementos válidos e inválidos.

```
input:not([type=range]):not([type=submit]):not([type=button]):not([type=checkbox]):invalid
```


```
{  
    background-image: url('../imgs/invalid.png');  
}
```

```
input:not([type=range]):not([type=submit]):not([type=button]):not([type=checkbox]):valid
```

```
{  
    background-image: url('../imgs/valid.png');  
}
```

Mi email es:  
(puedes introducir varios separados por comas)

*alex*



Mi email es:  
(puedes introducir varios separados por comas)

*alex@gmail.com*



## Aplicando estilos a campos requeridos o inválidos

Problema: aparecerá la imagen de **válido o inválido en todos los campos**, por lo tanto, la imagen de obligatorio no se mostrará.

Solución: sólo aplicar los estilos de campo válido o inválido cuando el elemento **tenga el foco** utilizando la pseudo-clase :focus.

```
input:not([type=range]):not([type=date]):not([type=submit]):not([type=button]):not([type=checkbox]):not([type=number]):focus:invalid {  
    background-image: url('../imgs/invalid.png');  
}  
  
input:not([type=range]):not([type=date]):not([type=submit]):not([type=button]):not([type=checkbox]):not([type=number]):focus:valid {  
    background-image: url('../imgs/valid.png');  
}
```

Debemos tener en cuenta que algunos navegadores, como por ejemplo Firefox, muestran **una sombra roja alrededor** de los elementos inválidos. Para evitarlo añadimos la siguiente regla css:

```
:invalid { box-shadow: none; }
```

## Estilos para el texto seleccionado

Con el selector `::selection`, podemos **cambiar el estilo al texto seleccionado**. Para que funcione en Firefox tendremos que utilizar también el prefijo correspondiente (`::-moz-selection`).

Por ejemplo, para cambiar los estilos del texto seleccionado de nuestro formulario de registro haremos lo siguiente:

```
::-moz-selection{ background: #484848; color:#fff; text-shadow: none; }  
::selection { background:#484848; color:#fff; text-shadow: none; }
```

Firefox aplicará estos estilos tanto al texto seleccionado de los campos del formulario como al texto seleccionado de la página, sin embargo, otros navegadores, como Chrome, sólo se los aplicarán al texto seleccionado de la página.

# Múltiples imágenes de fondo

Misma sintaxis que para colocar una sola imagen pero separando los valores de cada imagen individual con una coma. Por ejemplo:

background-image:

```
url(primerImagen.jpg),  
url(segundaImagen.gif),  
url(terceraImagen.png);
```

También, podemos utilizar el **atajo**\* para la propiedad background:

background:

```
url(primerImagen.jpg) no-repeat 0 0,  
url(segundaImagen.gif) no-repeat 100% 0,  
url(terceraImagen.png) no-repeat 50% 0;
```

**\*Atajo** de background.

```
body {  
  background-color: #ffffff;  
  background-image: url("img_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top;  
}
```



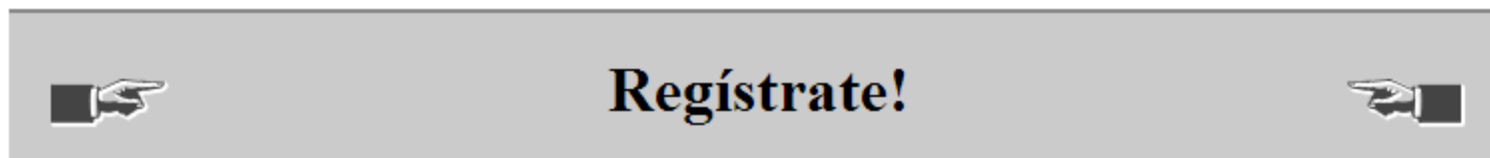
```
body {  
  background: #ffffff url("img_tree.png") no-repeat right top;  
}
```



# Múltiples imágenes de fondo

A continuación se muestra un ejemplo de cómo utilizar dos imágenes de fondo para el título de un formulario:

```
form h1 {  
    font-size: 2em;  
    padding-bottom: 20px;  
    background:  
        url(../imgs/bg-formtitle-left.png) left 13px no-repeat,  
        url(../imgs/bg-formtitle-right.png) right 13px no-repeat;  
}
```



## Redimensionar imágenes de fondo

Esta propiedad funciona con los siguientes prefijos para los diferentes navegadores:

- webkit-background-size
- moz-background-size
- background-size

Le indicaremos **dos valores: alto y ancho**, con cuidado ya que, podemos provocar que la imagen se distorsione si la relación entre el ancho y el alto no es adecuada.

Para evitar este problema podemos **indicar un único valor y el otro se calculará de forma automática** para mantener la relación de aspecto. Así, las siguientes declaraciones son equivalentes:

- background-size: 100px auto;
- background-size: 100px;

Si tuviéramos varias imágenes de fondo podríamos indicar sus tamaños **separando los valores por comas**, pero si sólo indicamos un tamaño, este se aplicaría para todas ellas:

- background-size: 100px auto, auto auto;

## Redimensionar imágenes de fondo

El tamaño predeterminado de la imagen de fondo es el **tamaño real** de la imagen. A veces, la imagen es sólo un poco más pequeña o más grande que el elemento que la contiene. En estos casos, nos puede interesar indicar que la imagen se **redimensione para cubrir todo el elemento**, esto lo haremos utilizando la palabra cover:

```
background-size: cover;
```

También podemos indicar el tamaño de la imagen en **porcentajes del elemento que la contiene**.

```
background-size: 50%;
```

## Fuentes incrustadas

La propiedad `@font-face` permite a los diseñadores vincular un archivo que contiene una **fuentes específica**.

Para incluir las fuentes mediante `@font-face`, haremos lo siguiente:

1. **Cargaremos el archivo** de fuente en nuestro sitio web en una variedad de formatos para que sea soportado por los diferentes navegadores.
2. Nombrar, describir y **enlazar** la fuente en una regla `@font-face`.
3. Incluir el **nombre de la fuente** en un valor de la propiedad `font-family`, tal como lo haríamos para las fuentes del sistema.

He aquí un ejemplo de un bloque de `@font-face`:

```
@font-face {  
    font-family: 'MiNuevaFuente';  
    src: url('font.ttf');  
}
```

## Fuentes incrustadas

Necesitaremos **una regla @font-face para cada fuente** que queramos incrustar en nuestra página.

Además, también necesitaremos **reglas diferentes para variaciones** de la misma fuente: thin, thick, italic, etc.

Podemos declarar diferentes src para una misma regla @font-face separándolas por comas:

```
@font-face {  
    font-family: 'LeagueGothicRegular';  
    src:url('../fonts/League_Gothic-webfont.eot') format('eot'),  
    url('../fonts/League_Gothic-webfont.woff') format('woff'),  
    url('../fonts/League_Gothic-webfont.ttf') format('truetype'),  
    url('../fonts/League_Gothic-webfont.svg') format('svg');  
}
```

Si el navegador no reconoce el primer formato de fuente lo intentará con el segundo y así sucesivamente.

# Fuentes incrustadas

También tenemos la posibilidad de indicar, opcionalmente, descriptores de propiedades en la definición de la fuente (font-style, font-variant, font-weight y otros) para que coincidan con la fuente que se está incrustando.

```
@font-face {  
  font-family: 'LeagueGothicRegular';  
  src: url('../fonts/League_Gothic-webfont.eot') format('eot'),  
  url('../fonts/League_Gothic-webfont.woff') format('woff'),  
  url('../fonts/League_Gothic-webfont.ttf') format('truetype'),  
  url('../fonts/League_Gothic-webfont.svg') format('svg');  
  font-weight: bold;  
  font-style: normal;  
}
```

Con esto no le estamos diciendo al navegador que ponga la fuente en negrita, lo que hacemos es indicar que la fuente que estamos **incrustando es la variedad negrita** de la fuente. Por este motivo, si queremos incrustar **diferentes variedades** de la misma fuente, tendremos que crear **una regla @font-face para cada una** de ellas.

# Google Web Fonts

Otra alternativa a la hora de usar fuentes personalizadas es el uso de un servicio que nos ofrece Google (**Google Web Fonts**).

Una vez localizada la fuente, tenemos la opción de:

1. **Descargarla y enlazarla** como se ha descrito anteriormente
2. Utilizarla usando el **elemento <link> dentro del <head>**. Esta opción no es recomendable ya que si queremos utilizar esta fuente en varios documentos html de nuestro sitio, tendremos que enlazarla con todos ellos.
3. **Importar la fuente a nuestra hoja de estilos** usando **@import** Con este método podremos utilizar la fuente en todas las páginas que tengan vinculada nuestra hoja de estilo

Opc.2

**<link>**    **@import**

```
<link href="https://fonts.googleapis.com/css2?family=Ranchers&
display=swap" rel="stylesheet">
```

DIW

Opc.3

**<link>**    **@import**

```
<style>
@import url('https://fonts.googleapis.com/css2?family=Ranchers
&display=swap');
</style>
```