HTML5

HTML5

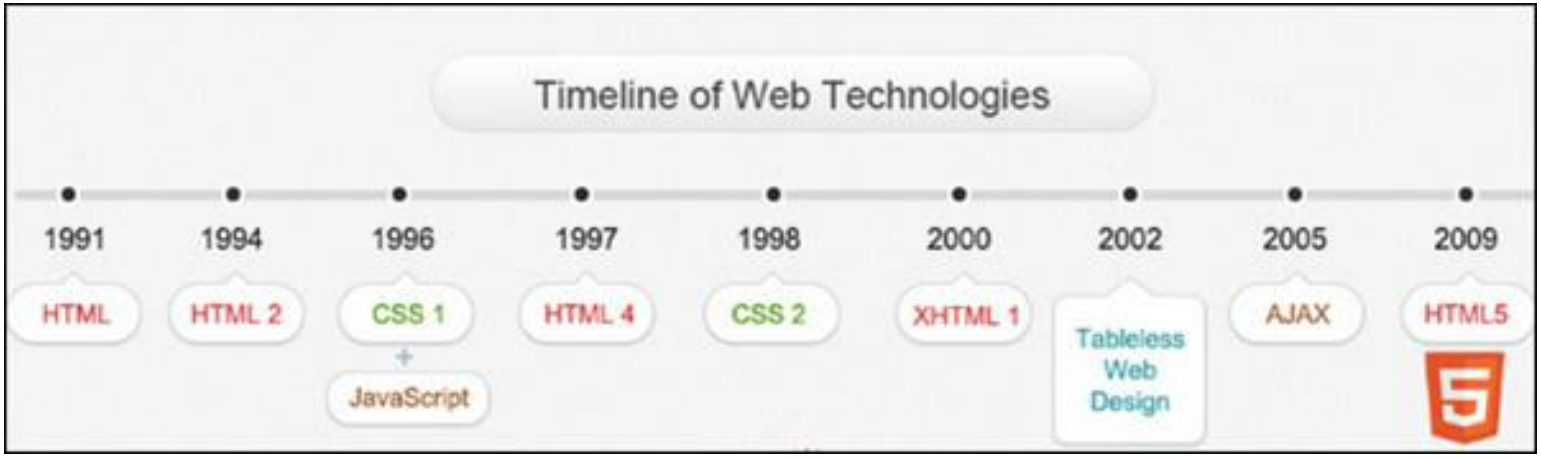
HTML es el lenguaje de marcado predominante y HTML5 es la última versión.

Incluye mejoras en las características existentes, nuevas características y APIs (interfaz de programación de aplicaciones) basadas en scripts.

API *(Application Programming Interface)* ➔ Conjunto de subrutinas, funciones y procedimientos (o métodos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción. Enmascara una compleja tarea de programación para ofrecer un servicio al usuario mucho más fácil de utilizar.

HTML5 incluye redefiniciones de elementos existentes, y también nuevos elementos que permiten a los diseñadores web ser más expresivos en la semántica de su código. El término HTML5, además, incluye una serie de otras nuevas tecnologías y APIs. Algunos de estas incluyen el dibujo, con el elemento <canvas>, almacenamiento fuera de línea, el nuevo <video> y <audio>, la función de arrastrar y soltar (drag-and-drop), fuentes incrustadas, y otros.

HTML5 es, de hecho, una mejora de la combinación entre Javascript, HTML y CSS. A partir de ahora, **HTML** provee los elementos estructurales, **CSS** se encuentra concentrado en cómo volver esa estructura utilizable y atractiva a la vista, y **Javascript** tiene todo el poder necesario para proveer dinamismo y construir aplicaciones web completamente funcionales.



Estructura de un documento

Los documentos HTML se encuentran estrictamente organizados por etiquetas específicas.

<!DOCTYPE>

En primer lugar necesitamos indicar el tipo de documento que estamos creando.

En contraposición con la definición extensa y compleja utilizada en anteriores versiones …

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Esto en HTML5 es extremadamente sencillo:

```
<!DOCTYPE html>
```

<html>

En HTML, la estructura tiene su raíz en el elemento <html> que envolverá al resto del código:

```
<!DOCTYPE html>
<html lang="es">
    …
</html>
```

El atributo **lang** en la etiqueta de apertura <html> es el único atributo que necesitamos especificar en HTML5.

<head> y <body>

El código HTML insertado entre las etiquetas <html> tiene que ser dividido entre dos secciones principales. La primera sección es la **cabecera** y la segunda el **cuerpo**. Estas secciones se crean en el código usando los elementos <head> y <body>.

El propósito de estas etiquetas sigue siendo exactamente el mismo que en versiones anteriores.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        …
    </head>
    <body>
        …
    </body>
</html>
```

<meta>

La nueva etiqueta que define el **juego de caracteres**, que especifica cómo el texto será presentado en pantalla, también conlleva una **simplificación**.

Anteriormente la declaración tenía el siguiente aspecto:

```
<meta http-equiv="content-type" content="text/html; charset=ISO-8859-1" />
```

Ahora, la definición del tipo de caracteres es más **corta y simple**.

Se pueden agregar otras etiquetas <meta> como ***description*** o ***keywords*** para definir otros aspectos de la página.

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="iso-8859-1">
        <meta name=”description” content=”Introducción HTML5”>
        <meta name=”keywords” content=”HTML5, CSS3, IES Pere Maria”>
    </head>
    ...
```

En HTML5 no es necesario cerrar etiquetas simples con una barra al final como sucedía en XHTML, pero es recomendable utilizarlas por razones de compatibilidad. El anterior código se podría escribir de la siguiente manera, en ambos casos sería HTML5 válido:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="iso-8859-1" />
        <meta name=”description” content=”Introducción HTML5” />
        <meta name=”keywords” content=”HTML5, CSS3, IES Pere Maria” />
        ...
    </head>
    <body>
        ...
    </body>
</html>
```

<title>

La etiqueta <title>, simplemente especifica el **título del documento**, que será visible en la pestanya del navegador:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="iso-8859-1" />
        <meta name=”description” content=”Introducción HTML5” />
        <meta name=”keywords” content=”HTML5, CSS3, IES Pere Maria” />
        <title>Diseño de interfaces Web</title>
        ...
    </head>
    <body>
        ...
    </body>
</html>
```

<link>

Este elemento se utiliza para incorporar estilos, imágenes o iconos desde **archivos externos**.
Uno de los usos más comunes para <link> es la incorporación de archivos con estilos CSS:

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="iso-8859-1" />
        <meta name=”description” content=”Introducción HTML5” />
        <meta name=”keywords” content=”HTML5, CSS3, IES Pere Maria” />
        <title>Diseño de interfaces Web</title>
        <link rel=”stylesheet” href=”estilos.css” />
        …
    </head>
    <body>
        …
    </body>
</html>
```

<script>

Este elemento se utiliza para incorporar **códigos Javascript** desde **archivos externos** o en el propio **cuerpo** del documento. Cambia con respecto a la versión anterior:

```
<script src="scripts.js" type="text/javascript"></script>
```

```
<!DOCTYPE html>
<html lang="es">
    <head>
        <meta charset="iso-8859-1" />
        <meta name=”description” content=”Introducción HTML5” />
        <meta name=”keywords” content=”HTML5, CSS3, IES Pere Maria” />
        <title>Diseño de interfaces Web</title>
        <link rel=”stylesheet” href=”estilos.css” />
        <script src="scripts.js"></script>
    </head>
    <body>
        …
    </body>
</html>
```

Estructura del cuerpo

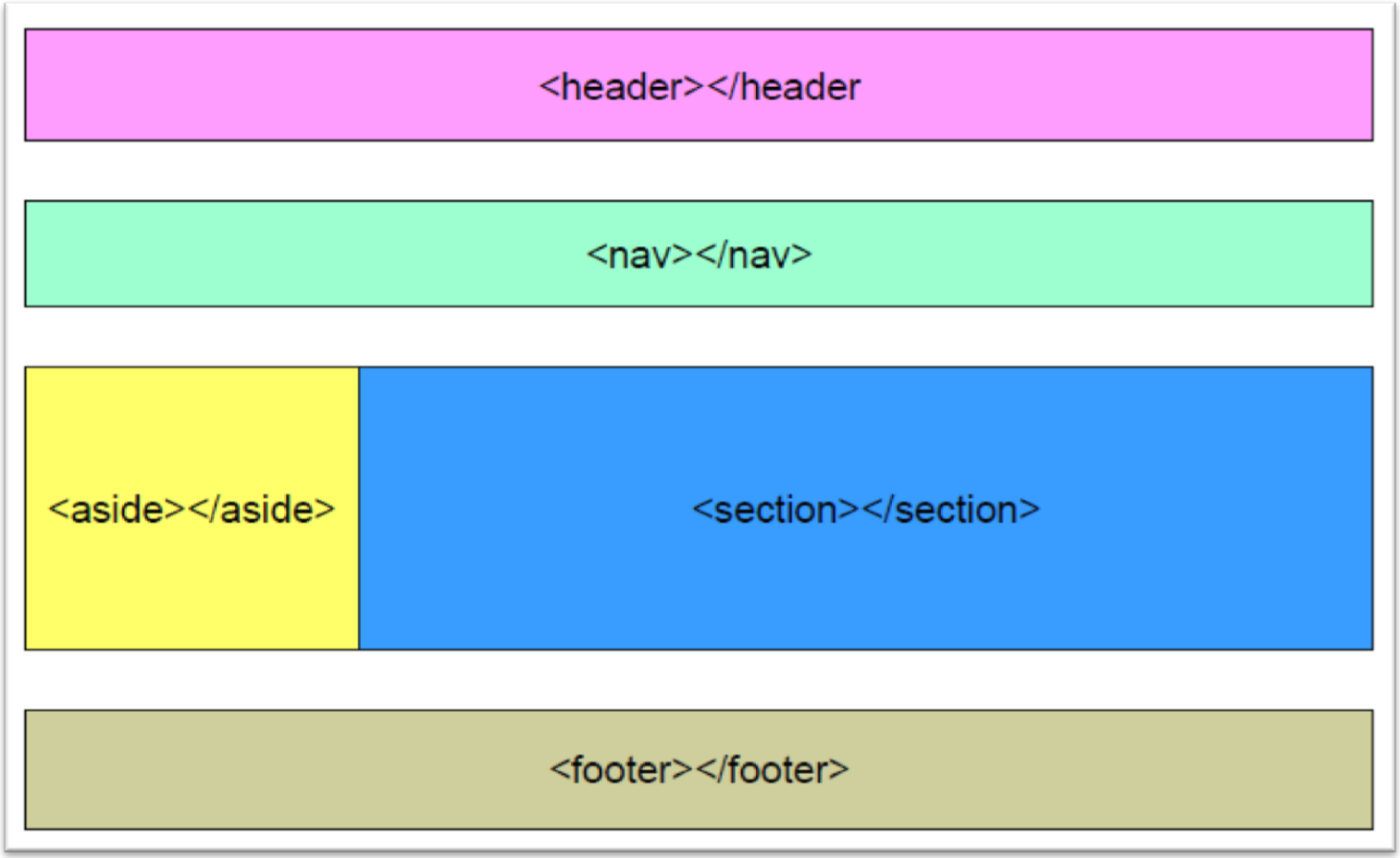
HTML ofrece diferentes formas de construir y organizar la información dentro del cuerpo de un documento. **En los inicios <table>** fue utilizado profusamente para estructurar el contenido, pero **su uso decayó con el tiempo y fue entonces el elemento ** el que comenzó a ser preponderante. Webs más interactivas y la integración de HTML, CSS y Javascript, generalizaron el uso de .

La palabra clave div solo especifica una división en el cuerpo, como la celda de una tabla, pero no ofrece indicio alguno sobre qué clase de división es, cuál es su propósito o qué contiene: ***No tiene una connotación semántica.***

Para los usuarios, estas claves o indicios no son importantes, pero para los navegadores, la correcta interpretación del interior del documento que se está procesando puede ser crucial en muchos casos. Tras la revolución de los dispositivos móviles y el surgimiento de diferentes formas en que la gente accede a la web, **la identificación de cada parte del documento es una tarea que se ha vuelto muy relevante**.

Las palabras claves que representan cada nuevo elemento de HTML5 están íntimamente relacionadas con estas áreas.

HTML5 considera esta estructura básica y provee **nuevos elementos** para diferenciar y declarar cada una de sus partes. A partir de ahora podemos decir al navegador para qué es cada sección:



El orden que elegimos para colocar los elementos HTML5 **depende de nosotros.**

HTML5 es versátil y simplemente nos proporciona los parámetros y elementos básicos con los que trabajar, pero cómo usarlos será exclusivamente decisión nuestra.

Un ejemplo de esta versatilidad es que el elemento **<nav>** podría ser insertado dentro del elemento <header> o en **cualquier otra parte del cuerpo**.

Sin embargo, siempre se debe considerar que estas etiquetas fueron creadas para brindar información a los navegadores y ayudar a cada nuevo programa y dispositivo en el mercado a identificar las partes más relevantes del documento.

Ejercicio: *Realiza un diseño flexible como el anterior utilizando las nuevas etiquetas html5.*

Elementos dentro del cuerpo

El contenido está compuesto por diferentes elementos como títulos, textos, imágenes, videos y aplicaciones interactivas, entre otros. Necesitamos poder **diferenciar estos elementos** y establecer una relación entre ellos dentro de la estructura.

<article>

Del mismo modo que los blogs están divididos en entradas, los sitios web normalmente presentan información relevante dividida en **partes que comparten similares características**. El elemento <article> nos permite identificar cada una de estas partes:

```
<header> …  </header>
<nav> … </nav>
<section>
     <article>
          Texto del mensaje I
     </article>
     <article>
          Texto del mensaje II
     </article>
</section>
<aside> … </aside>
```

Como una parte independiente del documento, el contenido de **cada elemento <article> tendrá su propia estructura**.

Para definir esta estructura, podemos aprovechar la versatilidad de los elementos **<header>** y **<footer>** estudiados anteriormente. Estos elementos son portables y pueden ser usados no solo para definir los límites del cuerpo sino también en cualquier sección de nuestro documento.

```
<article>
    <header>
        <h1>Título del mensaje I</h1>
        <h2>Subtítulo del mensaje I</h2>
        <p>publicado 20-09-2016</p>
    </header>
    Texto del mensaje I
    <footer>
        <p>comentarios (0)</p>
    </footer>
</article>
```

<figure> y <figcaption>

La etiqueta <figure> fue creada para ser aún más específicos a la hora de declarar el contenido del documento. Normalmente estos elementos son parte del contenido relevante pero pueden ser extraídos o movidos a otra parte sin afectar o interrumpir el flujo del documento, como **ilustraciones, fotos, videos, etc…**

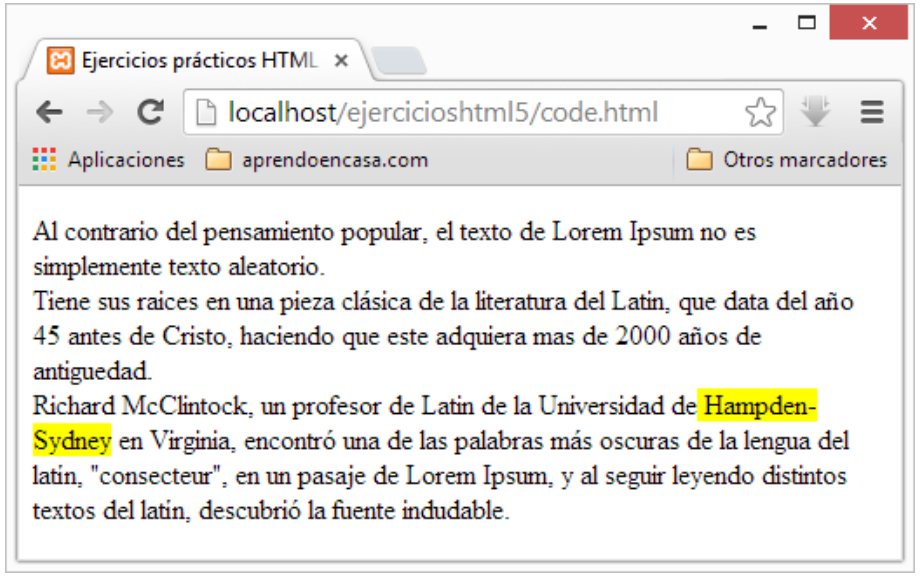
```
<article>
    <header>
        <h1>Título del mensaje I</h1>
        <h2>Subtítulo del mensaje I</h2>
        <p>publicado 20-09-2016</p>
    </header>
    Texto del mensaje I
    <figure>
        <img src=”./img/image.jpg”>
        <figcaption> Esta es la imagen del mensaje I </figcaption>
    </figure>
    <footer>
        <p>comentarios (0)</p>
    </footer>
</article>
```

Nuevos elementos y redefiniciones

<mark>

La etiqueta <mark> fue agregada para **resaltar parte de un texto**. El ejemplo que más se ajusta a este caso es un ***resultado de búsqueda***. El elemento <mark> resaltará la parte del texto que concuerda con el texto buscado:

```
<span>El <mark>ordenador</mark> está ocupado</span>
```



<small>

En HTML5, el nuevo propósito de <small> es **presentar la llamada letra pequeña**, como impresiones legales, descargos, etc…

```
<small>Derechos Reservados &copy; 2016 IES Pere Maria Orts</small>
```

<cite>

Otro elemento que ha cambiado su naturaleza para volverse más específico es <cite>. Ahora las etiquetas <cite> **encierran el título de un trabajo**, como un libro, una película, una canción, etc…

```
<span>Los contenidos del libro <cite>HTML5 CSS3, the Real World</cite></span>
```

<address>

El elemento <address> es un viejo elemento convertido en un **elemento estructural**. Podría ubicarse perfectamente en situaciones en que debamos presentar **información de contacto.**

```
<article>
    <header>
        <h1>Título del mensaje </h1>
    </header>
    Este es el texto del mensaje
    <footer>
        <address>
            <a href=”http://iesperemaria.edu.gva.es/”>IES Pere Maria Orts</a>
        </address>
    </footer>
</article>
```

<time>

En cada <article> de la última plantilla, podemos incluir la fecha indicando de cuándo se publicó el mensaje. Para esto podemos usar un simple elemento <p> dentro de la cabecera <header> del mensaje, pero existe un elemento en HTML5 específico para este propósito.

Permite declarar un texto comprensible para humanos y navegadores que representa **fecha y hora**.

```
<article>
    <header>
        <h1>Título del mensaje II</h1>
        <time datetime="2016-09-23" pubdate>Publicado 23-09-2016</time>
    </header>
    Texto del mensaje II
</article>
```

Ahora, el elemento <p> usado en ejemplos previos se ha reemplazado por el nuevo elemento <time> para mostrar la fecha en la que el mensaje fue publicado. El atributo **datetime** tiene el valor que representa la **fecha comprensible para el navegador** (timestamp).

El formato de este valor debe seguir un **patrón** similar al siguiente: **2016-09-23T12:43:02**.

El atributo **pubdate** indica que es la fecha de publicación del documento.

<progress>

Dos nuevos elementos en HTML5 permiten el marcaje de los datos que están siendo medidos: *progress y meter.*

El elemento **progress** se utiliza para describir el estado actual de un proceso de cambio que se está completando. La tradicional **barra de progreso de descarga** es un perfecto ejemplo. Puede tener un atributo ***max*** para indicar el punto en que la tarea estará completa, y un atributo ***value*** por indicar el estado de la tarea. Ambos atributos son opcionales.

```
<h1> Su tarea se está realizando </ h1>
<p>Estado:
    <progress min="0" max="100" value="0"> <span> 0 </span>% </progress>
</p>
```

Este elemento sería utilizado **mejor junto a un poco de JavaScript**, para cambiar dinámicamente el valor del porcentaje a medida que la tarea progresa.

<meter>

El elemento **meter**, por su parte, representa un elemento cuyo rango es conocido, lo que significa que tiene definido los valores mínimo y máximo.Tiene seis atributos asociados. Además de **max** y **value**, También permite el uso de **min**, **high**, **low** y **optimum**.

Los atributos **min** y **max** hacen referencia en los límites inferior y superior del rango, mientras que **value** indica la medida actual especificada. Los atributos **high** y **low** indican umbrales de lo que se considera "alto" o "bajo" en el contexto. **optimum** se refiere al valor ideal.

Por ejemplo, la calificación en un examen puede variar de 0% a 100% (max), por debajo del 60% se considera bajo y cualquier cosa por sobre del 85% se considera alto. En el caso de una puntuación de la prueba, el valor óptimo sería de 100.

```
<p>
    El uso total del disco actual: <meter value="63" min="0" max="320"
    low="10" high="300" optimum="100"> 63 GB </meter>
</p>
```

El elemento *details*

Este nuevo elemento ayuda a marcar una sección del documento que está oculta, pero puede ampliarse para revelar información adicional. El objetivo del elemento es proporcionar apoyo nativo para una característica común en la web, un **cuadro plegable que tiene un título**, y además información o funcionalidad escondidas.

```
<details>
    <summary>Ciclos de Informática </summary>
    <ul>
        <li> <cite> ASIX </ cite> </li>
        <li> <cite> DAW </ cite> </li>
        <li> <cite> DAM </ cite> </li>
    </ul>
</details>
```

En este ejemplo al hacer clic en ***summary***, el contenido oculto aparece. Si ***details*** no tiene un ***summary*** definido, el navegador definirá uno por defecto (por ejemplo, "Detalles"). **Si se desea que el contenido oculto sea visible por defecto**, se puede utilizar el atributo booleano ***open***.

El elemento ***summary*** solo se puede utilizar como un hijo de ***details***, y debe ser el primer hijo, si se utiliza. *No todos los navegadores soportan este elemento.*

El atributo async para Scripts

El elemento **`script`** permite ahora el uso del atributo **`async`**, similar al atributo **`defer`** existente.

Usando **`defer`** se indica que el navegador debe esperar hasta que todo el marcaje de la página se cargue antes de cargar el script.

El atributo **`async`** permite especificar que un script cargue de manera **asíncrona**, es decir, que cargue **tan pronto como esté disponible**, sin causar el retraso de otros elementos de la página.

Tanto **`defer`** como **`async`** son atributos booleanos.

```
<script src="demo_async.js" async></script>
```

Nuevos elementos para formularios

Hay cuatro nuevos elementos: **`output`, `keygen`, `progress`** y **`meter`.** Ya hemos visto los dos últimos anteriormente, por ser a menudo útiles fuera de los formularios. El elemento **keygen** ha quedado obsoleto, así que veremos únicamente el elemento **output**.

El elemento *output*

El propósito del elemento `output` es aceptar y **mostrar el resultado de un cálculo**. Debe ser utilizado cuando el usuario puede ver el valor, pero no manipularlo directamente, o cuando el valor se puede derivar otros valores introducidos en el formulario. Un ejemplo de uso podría ser el coste total calculado después de envío y los impuestos en un carrito de compra.

Normalmente, tendrá sentido utilizar *JavaScript* en el navegador para actualizar este valor. El elemento `output` tiene un atributo, que se utiliza para hacer referencia a los identificadores (atributos **`id`**) de los campos del formulario que están involucrados en el cálculo del elemento `output`. El nombre y el valor del elemento `output` se envían junto al formulario.

https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_output

Cambios en controles de formulario existentes

Algunos son los cambios en controles de formulario en HTML5:

El elemento *form*

Como hemos visto, HTML5 permite la validación de forma nativa en varios de los campos de formulario. Sin embargo, **podemos impedir que el formulario sea validado**. El nuevo atributo booleano **novalidate** permite enviar un formulario **sin validación nativa de sus campos**.

Asimismo, **los formularios ya no necesitan tener el atributo *`action`* definido**. Si se omite, el formulario se comportará como si la acción se asignara a la página actual.

Por último, el atributo **autocomplete**, introducido anteriormente, también **se puede añadir directamente al elemento `form`** en este caso, se aplicará a todos los campos del formulario.

HTML5 nos proporciona varios **atributos** que nos permiten **validar** lo que es un valor aceptable, e informar al usuario de los errores, todo **sin el uso de scripts**.

El atributo *required*

Si un campo obligatorio está vacío o no válido, el envío del formulario fallará, y el foco pasará al primer elemento inválido. La sintaxis es simplemente `required` o bien `required="required"` utilizando la sintaxis XHTML.

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

Se puede dar estilo a los atributos `required` con la pseudoclase `:required`. También se puede dar estilo en los campos válidos o inválidos con las pseudoclases `:valid` y `:invalid`.

```
input:required {  background-image:url ('../img/required.png');  }
input:focus:invalid {  background-image:url ('../img/invalid.png');  }
input:focus:valid {  background-image:url ('../img/valid.png');  }
```

El atributo *placeholder*

El atributo `placeholder` permite una breve descripción indicando al usuario qué datos se deben introducir en este campo. El texto de `placeholder` desaparece cuando se introduce algún dato, y vuelve a aparecer si se deja vacío.

```
<label for="url">Mi sitio web se encuentra en:</label>
<input type="text" id="url" name="url" placeholder="http://example.com" />
```

El atributo *pattern*

El atributo `pattern` permite proporcionar una expresión regular con la que la entrada del usuario debe coincidir para ser considerada válida.

Ejemplo: que la contraseña sea por lo menos de seis caracteres de largo y sin espacios.

```
<label for="password"> Contraseña: </label>
<input type="password" id="pass" name="pass" required pattern="\S{6,}" />
```

Ejemplo: que el campo contenga solamente 3 letras.

```
<input type="text" name="ccode" pattern="[A-Za-z]{3}" title="Country …" />
```

El atributo *disabled*

Los elementos de formulario con el atributo `disabled` **tienen su contenido en gris** en el navegador y el texto es más claro. Los navegadores no permitirán al usuario obtener el foco de un elemento de formulario con este atributo.

Se puede utilizar para *desactivar* el botón de envío hasta que todos los campos se llenen correctamente, por ejemplo. **Con la pseudoclase** `:disabled` de CSS **se puede cambiar este aspecto a los elementos deshabilitados** (`disabled`).

El atributo *readonly*

El atributo `readonly` es similar al atributo anterior. **Hace imposible el usuario modificar el campo** del formulario. Pero a diferencia de `disabled`, **el campo puede recibir el foco, y su valor sí que se envía con el formulario.**

En un formulario de comentarios, por ejemplo, es posible que se desee incluir la URL de la página actual o el título del artículo que se ha comentado, dejando que el usuario sepa que estamos recogiendo estos datos sin dejar que los cambien:

```
<label for="about">Título del artículo</label>
<input type="text" name="about" id="about" value="HTML5" readonly />
```

El atributo *multiple*

El atributo booleano **`multiple`** indica que varios valores se pueden introducir en un control de formulario. Se puede añadir a los **`select, input, email`** y **`file`**.

Ahora el usuario podrá seleccionar más de un archivo, o incluir varias direcciones de correo electrónico separadas por comas.

El atributo *form*

No confundir con el elemento **`form`**. El atributo **`form`** en HTML5 permite **asociar elementos de formulario** con formularios en los que no están anidados. Eso quiere decir que ahora se puede asociar un control **`fieldset`** u otro elemento de formulario con cualquier otro formulario en el documento.

```
<form action="./action_page.php" method="get" id="form1">
  <input type="text" name="fav_color"><br>
  <input type="submit">
</form>

<fieldset form="form1">
  Name: <input type="text" name="username"><br>
  Email: <input type="text" name="usermail"><br>
</fieldset>
```

El atributo *autocomplete*

Este atributo especifica si el formulario o un control de formulario, debe tener **la funcionalidad de completado automático**.

Para la mayoría de los campos del formulario, este será un desplegable que aparece cuando el usuario empieza a escribir. Para los campos de contraseña, es la capacidad de guardar la contraseña al navegador.

Por defecto, la función de llenado automático está activada. Para desactivarla, se debe utilizar `autocomplete="off",` que es una buena idea para información confidencial, como el número de tarjeta de crédito, o información que nunca se reutiliza, como un *captcha*.

El atributo `autocomplete` también está controlado por el navegador. El usuario deberá activar la funcionalidad de autocompletar en su navegador para que funcione.

```
<form id="registro" method="post" autocomplete />
<input type="password" id="password" name="password" autocomplete="off" />
```

Los atributos *list y datalist*

El objetivo es crear un *campo de texto con conjunto predefinido de opciones* para autocompletar. A diferencia del elemento **select**, *el usuario puede introducir cualquier dato que desee*, aunque se le presentará un conjunto de opciones sugeridas en un desplegable a medida que escriba.

La lista de opciones se crea con el elemento **datalist**, que después se le asocia a un **input**. El atributo **list** toma como valor del atributo **id** del **datalist** que se desea asociar.

```
<label for="favcolor">Color favorito</label>
<input type="text" list="colores" id="favcolor" name="favcolor">

<datalist id="colores">
    <option value="Azul">
    <option value="Verde">
    <option value="Rosa">
    <option value="Rojo">
</datalist>
```

El atributo *autofocus*

El atributo booleano **autofocus** especifica que un control de formulario deberá recibir el foco tan pronto como se cargue la página. Solo un elemento de formulario puede tener el enfoque automático en una página determinada. Ejemplo:

```
<li>
   <label for="regnombre">Mi nombre es:</label>
   <input type="text" id="regnombre" name="regnombre" required autofocus />
</li>
```

Nuevos elementos input

Estos son los tipos de `input` que existían antes de HTML5: **button, checkbox, file, hidden, image, password, radio, reset, submit, text.**

HTML5 añade 13 tipos de entrada que ofrecen más especificidad de datos y de validación: **search, email**, **url**, **tel**, **datetime**, **date**, **month**, **week**, **time**, **datetime-local**, **number**, **range**, **color**

search

El tipo de input de tipo `search` ofrece un campo de búsqueda: un campo `input` de texto de una sola línea para introducir uno o más términos de búsqueda.

La diferencia entre el campo de texto y el campo de búsqueda es principalmente **estilística**. El nuevo tipo de búsqueda es una señal visual para indicar al usuario donde debe ir a buscar en el sitio web, y proporciona una interfaz a la que el usuario está acostumbrado.

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

email

El tipo **email** es utilizado por especificar direcciones de correo electrónico. Es compatible con el atributo booleano múltiple, que permite múltiples valores separados por comas.

```
<label for="email">Mi dirección de correo electrónico es:</label>
<input type="email" id="email" name="email">
```

Se diferencia de un campo de texto sin formato en que si se mete el foco en el campo de correo electrónico, los dispositivos mostrarán un teclado optimizado para la entrada de correo electrónico (con una tecla de acceso directo para el símbolo @).

URL

La entrada de tipo **url** se utiliza para especificar una dirección web. Igual que el correo electrónico, se mostrará como un campo de texto normal. En muchas pantallas táctiles, el teclado en pantalla visualizado será optimizado para la entrada de dirección web, con una barra (/) y una tecla de acceso directo ".com".

El navegador considerará la entrada como inválida si la URL tiene un formato incorrecto. Si se desea que el valor a ingresar se ajuste a un formato más específico, se puede utilizar el atributo **pattern** para asegurar que el patrón es correcto.

números de teléfono

Para números telefónicos, se utiliza `type = "tel".` A diferencia de los tipos **`url`** y **`email`**, el tipo de teléfono no hace cumplir una sintaxis o patrón particular. Se puede sugerir un formato particular mediante la inclusión de un **`placeholder`** con la sintaxis correcta. Se puede estipular un formato mediante el atributo **`pattern`**.

```
<li>
    <label for="url">Mi teléfono es:</label>
    <input type="tel" id="tel" name="tel" placeholder="965.555.555"
    pattern="[0-9]{3}.[0-9]{3}.[0-9]{3}">
</li>
```

number

El tipo **number** proporciona una entrada para introducir un número. Se puede introducir un número o hacer clic a flechas hacia arriba o hacia abajo para seleccionar un número.

```
<label for="cantidad">Querría recibir <input type="number" name="cantidad"
id="cantidad" min="0" max="10" step="1"> copias del Periódico HTML5</label>
```

range

El tipo de entrada **rango** muestra un control deslizante en el navegador.

```
<label for="escala">Mi conocimiento de HTML5 (1-10):</label>
<input type="range" min="1" max="10" name="escala" type="range">
```

color

El tipo de entrada de color proporciona al usuario un selector de color. El selector de color debe devolver un valor de color RGB hexadecimal, *como #FF3300*.

```
<label for="favcolor">Color favorito</label>
<input type="color" id="favcolor" name="favcolor">
```

Se puede proporcionar un **`placeholder`** indicando que se requiere un formato RGB hexadecimal, y utilizar el atributo **`pattern`** para restringir la entrada solo de valores de color hexadecimal.

```
<label for="clr">Color: </label>
<input id="clr" name="clr" type="text" placeholder="#FFFFFF"
pattern="#(?:[0-9A-Fa-f]{6}|[0-9A-Fa-f]{3})" required>
```

fechas y horas

Hay diversos nuevos tipos de **`input`** de fecha y hora, mostrados a continuación:

- **date** comprende la fecha (año, mes y día):

```
<label for="fechainicio">Iniciar mi suscripción:</label>
<input type="date" min="2015-03-17" max="2015-05-17" id="fechaini"
name="fechaini" required placeholder="aaaa-mm-dd">
```

- **month** incluye el año y el mes: `2012-12`
- **week** cubre el año y el número de la semana (de 1 a 52): `2011-W51`
- **time** hora del día, formato militar (reloj de 24 horas): `22:00`
- **datetime** incluye tanto la fecha y hora, separadas por una "T", y seguido por una zona horaria especificada: `2015-03-17T10:45-5:00` *representa 10:45 el 17 de marzo del 2015, en la zona horaria UTC menos 5 horas.*
- **datetime-local** es idéntica a ***`datetime`***, excepto que se omite la zona horaria.