DESARROLLO DE APLICACIONES WEB / MULTIPLATAFORMA

UD 2 Utilización de los lenguajes de marcas en entornos web

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

JOSÉ HIPÓLITO BOGAS PERONA

Indice

0.- Objetivos

1. Descubrir la historia y la evolución de HTML .
2. Presentar la estructura y la sintaxis de los documentos HTML.
3. Conocer los elementos básicos del lenguaje .
4. Profundizar en el conocimiento de las distintas etiquetas avanzadas, su significado y uso.
5. Aprender a construir documentos basados en etiquetado semántico.
6. Desarrollar los mecanismos de maquetación y el comportamiento de los tipos de elementos.
7. Comparar el lenguaje HTML con XHTML, su equivalente en lenguaje XML.
8. Enumerar algunas de las herramientas disponibles para visualizar y editar documentos HTML y XHTML.

1.- Introducción

La *World Wide Web,* más conocida como *web,* es el servicio de la red de Internet más popular. Su principal objetivo es permitir la distribución de documentos entre personas con independencia de la plataforma o de sistema operativo que utilicen. La *web* es, junto con el correo electrónico, uno de los servicios que han hecho de *Internet* la red global que es.

HTML (junto con XHTML, su versión XML) es el lenguaje que se utiliza para dar forma a los documentos publicados en la web. Es gratuito, sencillo, potente y no requiere de ningún software sofisticado para su edición, ya que un editor de texto es suficiente para crear cualquier tipo de documento.

2.1.- Introducción, evolución y estado actual de HTML

HTML es, sin ninguna duda, el lenguaje de marcas más conocido y utilizado. Sus siglas significan en castellano «lenguaje de marcado de hipertexto» HyperText Markup Language) y es el lenguaje en el que están escritas la gran mayoría de páginas de la web. Obviamente, todos los navegadores web reconocen HTML y lo saben interpretar y traducir

generando la representación de la página web tal y como la vemos los usuarios. El lenguaje HTML y la *web* son términos que van necesariamente de la mano, ya que de manera independiente no habrían tenido la repercusión que han conseguido conjuntamente.

En la actualidad HTML está gestionado por el W3C (World Wide Web Consortium), un consorcio dedicado a la estandarización de la web para lograr que ésta sea accesible, universal, fiable, fácil de usar y de uso libre. **HTML es, por lo tanto, gratuito.** Ninguna persona ni empresa debe pagar por utilizarlo.

«La web es uno de los servicios de Internet, pero no el único; el correo electrónico y la transferencia de archivos son algunos de los más utilizados. Existen muchos más servicios, algunos en desuso, como Gopher o Telnet.»

En 1991 Tim Berners-Lee diseñó HTML como el lenguaje de marcado capaz de proporcionar un mecanismo sencillo, flexible y universal para almacenar y transmitir información entre distintos sistemas informáticos. Aunque las redes de ordenadores ya existían desde muchos años antes, Internet por proponía una red global de comunicaciones y había que dotarla de soluciones para que las personas que las fuesen a utilizar pudiesen compartir información fácilmente. Ese primer diseño era muy simple incluía 18 etiquetas, algunas de las cuales aún se mantienen en la versión 5 de HTML. Difícilmente Tim Berners-Lee pudo anticipar el éxito de su creación, lo cual no le quita ningún mérito. HTML está bien diseñado, es sólido y se fundamenta en unas bases acertadas.

Existen dos hechos fundamentales que han marcado el destino de HTML:

- ➔ Los destinatarios de HTML han sido desde el principio **las personas**.
- ➔ La información proporcionada por HTML es de **naturaleza estética** y no semántica.

Estas dos características tienen algunas consecuencias. Una de ellas es que los navegadores, al procesar el Código HTML, si encuentran algún error de sintaxis, pueden intentar solucionarlo, y si la solución propuesta no es correcta, los resultados no serán catastróficos. Las personas tenemos una capacidad de interpretación de la información mucho más flexible que las máquinas, por lo que, ante un error de interpretación por parte del navegador, el resultado nos resultará inconsistente o poco atractivo estéticamente, pero seguro que seremos capaces de interpretarlo de forma correcta. La semántica la aporta al usuario en función de lo que ve y en y de su buen criterio. Si la página no está bien maquetada, el autor de ésta se dará cuenta visualmente del error. Si no lo percibe, es que el error no es relevante y las consecuencias no serán graves.

«Para abrir con un navegador web un documento HTML almacenado en el sistema de archivos local en el sistema operativo Windows, hay que pulsar con el botón derecho del ratón sobre fichero, seleccionar la opción **Abrir con** y elegir el navegador deseado.»

Por otro lado, si es un programa informático el que tiene que analizar y procesar el texto, éste ha de ser riguroso y carente de errores sintácticos. Un programa de ordenador es determinista (ante las mismas entradas debe generar las mismas salidas) y esto implica que no dispone de la capacidad de **interpretar** en el sentido humano.

Es por estas circunstancias por las que los navegadores web son laxos en la interpretación del Código HTML y es una de las razones de la popularidad y alcance de la web. Si hacer páginas HTML fuese muy difícil, su aceptación habría sido probablemente más lenta. Por suerte, los diseñadores y programadores de los navegadores entendieron que el proceso de escribir una página web no puede ser igual de estricto que el proceso de escribir un programa informático en cualquier lenguaje de programación.

HTML, desde su nacimiento, ha pasado por diversas revisiones que han ido ampliando y mejorando la capacidad del lenguaje.

En la siguiente tabla se muestran las diferentes versiones de HTML y su años de publicación.

Tabla 2.1. Evolución de las versiones de HTML

Año	Versión	Observaciones
1991	Diseño inicial	Incluye 18 etiquetas. Se considera como un dialecto de SGML. Se plasma en un documento interno e informal del CERN (Conseil Européen pour la Recherche Nucléaire u Organización Europea para la Investigación Nuclear, en español).
1992	HTML 1.1	Primer borrador informal.
1995	HTML 2.0	Publicado por el IETF (Internet Engineering Task Force o Grupo de Trabajo de Ingeniería de Internet, en español).
1997	HTML 3.2	Publicado como recomendación del W3C (World Wide Web Consortium).
1997	HTML 4.0	Publicado como recomendación del W3C.
1999	HTML 4.1	Publicado como recomendación del W3C.
2000	XHTML	Publicado como recomendación del W3C. Tenía como objetivo sustituir a HTML. En HTML5 se incorpora como parte de la recomendación y deja de desarrollarse por separado.
2014	HTML 5.0	Publicado como recomendación del W3C.
2016	HTML 5.1	Publicado como recomendación del W3C.
2017	HTML 5.2	Publicado como recomendación del W3C.

Cada versión mejora a la anterior. Incorpora nuevas etiquetas, elimina otras obsoletas y se acomoda las capacidades tecnológicas. Hay que tener en cuenta que el avance en las tecnologías relacionadas con la informática y las telecomunicaciones es de gran importancia y se da en cortos periodos de tiempo.

Por ejemplo, en los primeros años de existencia de Internet, el acceso a la red se hacía a través de la línea telefónica analógica mediante el uso de un **módem** (conversor de señales digitales-analógicas y analógicas-digitales) a una velocidad impensable para la velocidad de las línea digitales actuales. Se ha pasado en 30 años de limitar el número de bytes que tiene una fotografía para agilizar la carga de la página a transmitir vídeo en *streaming* en alta definición.

HTML ha ido adaptándose con éxito a estos avances para seguir siendo el lenguaje estándar de marcado para la difusión de contenidos a través de la web, aceptado por toda la industria tecnológica sin objeciones

La versión 5 (y posteriores) de HTML ha recogido las necesidades de un mercado dinámico y vertiginoso y ha abierto el nuevo e importante conjunto de oportunidades gracias a su potencial y a su capacidad, respetando los principios de su fundación.

Actividad propuesta 2.1

2.2.- Estructura y sintaxis del lenguaje HTML

El lenguaje de marcado HTML define el contenido de una página web. Los navegadores interpretan las marcas contenidas en los documentos HTML y representan la información para que los usuarios la puedan consultar e interactuar con ella. Trabajando junto con las hojas de estilo CSS, HTML es un increíble lenguaje de representación de información.

El lenguaje HTML ha tenido un enorme éxito debido en parte a su sencilla estructura y sintaxis.

2.2.1.- Estructura y sintaxis

Los elementos que forman las páginas HTML están identificados por marcas o etiquetas. Estas marcas «envuelven» el contenido de la página ya sea este texto, imágenes o cualquier otro tipo de elemento. Las marcas o etiquetas están compuestas por un nombre rodeado de los símbolos “<” y >”. **<etiqueta>**

El nombre de la etiqueta puede estar escrito en mayúsculas o minúsculas, ya que no se distingue entre ambos tipos de letra. No obstante, se recomienda que el etiquetado sea uniforme y preferiblemente en minúsculas.

Existen dos tipos principales de etiquetas: las etiquetas apertura y las de cierre. Un contenido debe estar rodeado por ambas etiquetas. La etiqueta de apertura se incluye antes del contenido y la etiqueta de cierre después de éste. La etiqueta de apertura se diferencia de la etiqueta de cierre en que esta última incluye el símbolo /” justo antes del nombre. **</etiqueta>**

Ambas etiquetas tendrán necesariamente el mismo nombre y rodearán un contenido. El conjunto formado por el **contenido** y las etiquetas de **apertura** y **cierre** forman un **elemento.** **<etiqueta>Esto es el contenido.</etiqueta>**

En HTML no todas las etiquetas necesitan apertura y cierre, ya que no todas son contenedoras. Un ejemplo es la **etiqueta “<br>”**. Esta etiqueta provoca un salto de línea en el texto, pero no debe rodear contenido, ya que el salto de línea es independiente del contexto. Por lo tanto, algunos elementos solo estarán formados por una etiqueta de apertura. A estos elementos se les denomina **elementos vacíos**. **“
”**

Dentro de un elemento se pueden incluir atributos. Los atributos permiten caracterizar Elemento de cara a identificarlo, representarlo o tratarlo de manera específica. La existencia de atributos es opcional y su número puede variar. Todos los elementos de un documento HTML pueden disponer de atributos, pero no todos los atributos son aplicables a cualquier elemento.

Los atributos se incluyen en la etiqueta de apertura y están compuestos por un nombre, el símbolo "=" y el valor del atributo entrecomillados. Las comidas pueden ser simples o dobles. De nuevo, se recomienda que el uso de comillas simples o dobles sea homogéneo a lo largo del documento, preferiblemente dobles.

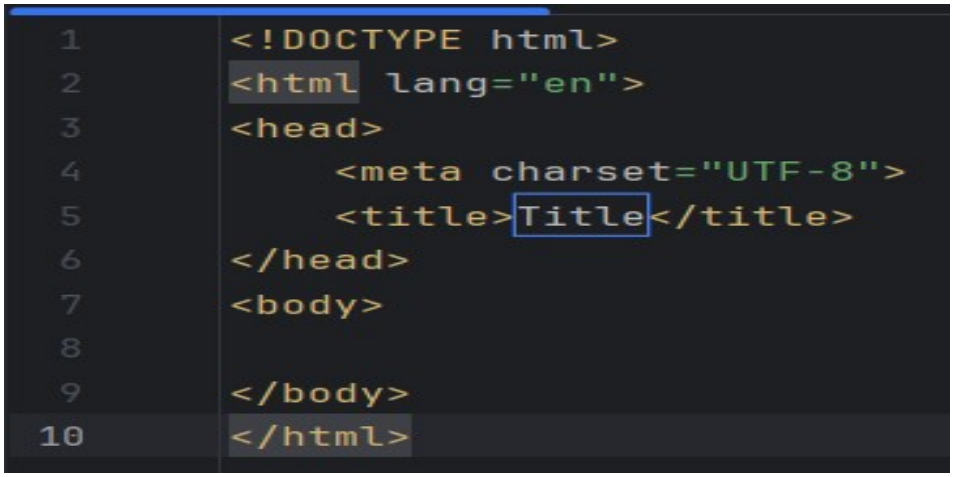
<etiqueta atributo1="valor1" atributo2="valor2">Esto es el contenido.</etiqueta>

Los elementos del documento HTML se pueden anidar Un elemento puede contener a 1 o más elementos en su interior. La anidación debe estar bien construida. No se permite que los elementos se solapen En el siguiente código se muestran dos ejemplos de anidación.

```
<elementonivel1>
        <elementonivel2>
                <elementonivel3>
                        Contenido
                </elementonivel3>
        </elementonivel2>
</elementonivel1>
```

```
<elementonivel1>
        <elementonivel2>
                <elementonivel3>
                        Contenido
        </elementonivel2>
                </elementonivel3>
</elementonivel1>
```

Una página HTML está compuesta por un conjunto de elementos y alguna información adicional que se indicará a continuación Este conjunto de elementos debe respetar una estructura definida en el estándar de HTML En el siguiente código se muestra la estructura básica de una página HTML:



El primer elemento (línea 1) identifica el tipo del documento. Este elemento es un elemento vacío y por lo tanto, no requiere de etiqueta de cierre. Aunque no es exactamente un elemento de HTML se incluyen las páginas HTML para indicar al navegador de qué tipo es el documento que va a procesar.

El siguiente elemento (entre las líneas 2 y 10) se define con la etiqueta <html>. Es el elemento raíz y dentro de él se encuentra todo el contenido de la página.

El elemento etiquetado como <head> (entre las líneas 3 y 6) incluye información no visible en la representación de la página dentro del navegador pero que sí afecta a cómo está va a ser presentada. En esta sección se indica, por ejemplo, el título que aparece en la pestaña del navegador, la hoja de estilo CSS que utilizar o el juego de caracteres que se usará para la representación de los textos.

Por último el elemento <body> (entre las líneas 6 y 8) albergará el contenido visible de la página, ya sea este texto, tablas, lista,s imágenes o elementos multimedia por citar algunos ejemplos.

Actividad resuelta 2.1

2.2.2.- Entidades

Algunos caracteres tienen un significado especial en HTML, son invisibles o, sencillamente, no existen en el lenguaje en el que está escrita la página. Por ejemplo, el símbolo «menor que» (<) es el comienzo de una etiqueta, los espacios en blanco múltiples son ignorados o las tildes no existen en algunos idiomas. Las entidades

permiten indicar al navegador que debe representar un carácter determinado solucionando los problemas derivados de estas circunstancias.

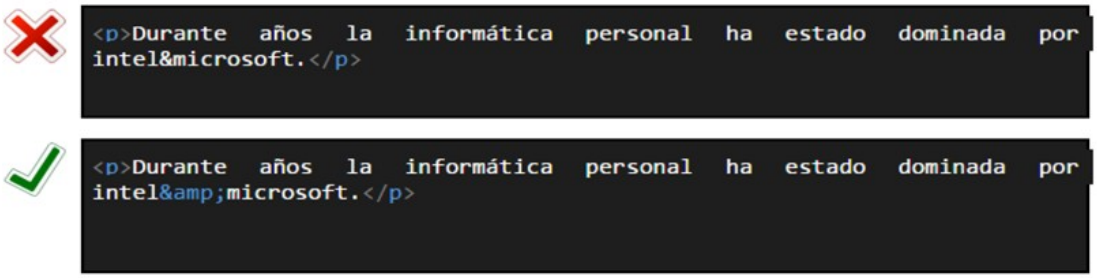
Por ejemplo, utilizando el juego de caracteres ISO-80859-1, el texto **¡Soy un campeón!** se muestra en el navegador como **Â¡soy un campeÂ³n!**. Como se puede observar el símbolo y el carácter no se han visualizado correctamente. Aunque este problema se podría resolver utilizando el juego de caracteres UTF-8, también se podría solucionar sustituyendo los caracteres problemáticos por su entidad equivalente: el carácter «¡» se sustituye por la entidad **¡** y el carácter «ó» por **ó**. El texto en el documento HTML quedaría de la siguiente manera: **¡Soy un campeón!**.

Otro caso podría ser cuando tenemos que dejar espacios en blanco que como hemos comentado el navegador no los respeta. Si queremos que los respete hemos de utilizar la entidad: ** **.

Una entidad en HTML es un conjunto de caracteres que permite indicar al navegador que represente caracteres que, por diversas razones, no se representan correctamente. Comienza por el símbolo «**&**» y termina con el símbolo «**;**»

Algunas de las entidades más comunes se pueden consultar en la siguiente tabla:

Carácter	Nombre	Entidad de carácter	Entidad numérica	Recomendación	Descripción
&	amp	&amp;	&#x26;	HTML 2.0	et (en inglés, ampersand)
<	lt	&lt;	&#x3C;	HTML 2.0	menor
>	gt	&gt;	&#x3E;	HTML 2.0	mayor
"	quot	&quot;	&#x22;	HTML 2.0	comillas
'	apos	&apos;	&#x27;	XHTML 1.0	apóstrofo



Se puede consultar la vista completa de entidades en la lista oficial:

https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references

2.2.3.- Comentarios

HTML admite incluir **líneas de comentarios**. Estos elementos no tienen ningún efecto sobre la página, ya que su único objetivo es proporcionar información a los diseñadores y programadores cuando están consultando el código interno de ésta. El contenido se

delimita con los caracteres **<!--** y **-->** y será ignorado por el navegador. A continuación, se muestra un ejemplo de un comentario incrustado en un bloque de código HTML.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Mi primera página</title>
</head>
<body>
    <!-- Esta es mi primera página HTML-->
    <h1>¡HOLA</h1>
</body>
</html>
```

2.3.- Elementos de HTML

Un documento HTML está compuesto de elementos organizados de manera jerárquica. El elemento <html> es la raíz y todos los demás deciden directa o indirectamente de él.

Como se han indicado anteriormente los elementos <html>, <head> y <body> son fundamentales dentro de la construcción de la página para indicar que el documento es HTML (elemento <html>), cómo se ha de representa la información (elemento <head> y qué información se debe presentar (elemento <body>), pero sólo son la punta de un enorme iceberg.

La relación de elementos de HTML es muy extensa y admite múltiples clasificaciones. En esta sección se muestran los elementos organizados por familias en función de su funcionalidad. En algunos casos, debido a su singularidad y relevancia, se muestran elementos de manera individual y no agrupados. Es necesario indicar que la relación no es exhaustiva, ya que el lenguaje es muy amplio y el objetivo es ser didáctico y no un manual de referencia.

Cada elemento se identifica por una etiqueta y se caracteriza por un conjunto de atributos, muchos de los cuales son específicos del elemento en concreto. No obstante, existen algunos atributos globales que es importante conocer, ya que son determinantes para el concreto funcionamiento de tecnologías relacionadas con HTML como las hojas de estilo CSSO el lenguaje de programación Javascript. Los atribulos globales son comunes a todos los elementos HTML.

En la actualidad, el lenguaje utilizado para programar embebidos en los documentos HTML es Javascript, en concreto un dialecto de la especificación ECMAScript.

En la siguiente tabla se detallan algunos de los atributos globales más utilizados:

Tabla 2.3. Los atributos globales son compartidos por todos los elementos

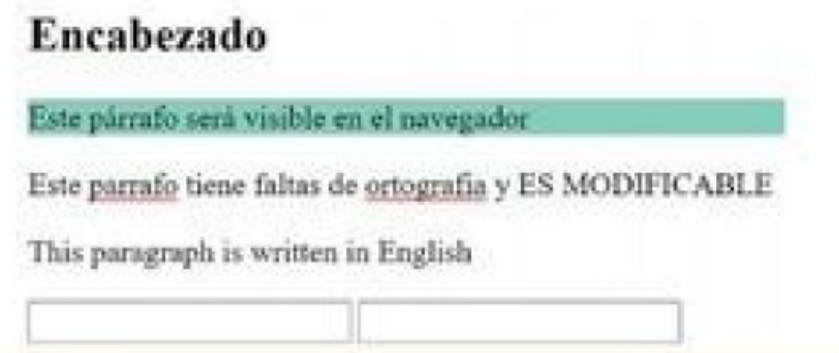
Atributo	Descripción
class	Permite agrupar elementos. El valor de este atributo es una cadena de caracteres definida por el autor de la página. Si varios elementos comparten el mismo valor, se dice que son de la misma clase. Se utiliza para asignar estilos y durante la programación con JavaScript.
contenteditable	Si existe, el contenido del elemento es editable.
hidden	Si existe, el elemento no se representa en el navegador.
id	Identifica al elemento de manera única en el documento. El valor es una cadena de caracteres definida por el autor de la página. Se utiliza para asignar estilos y durante la programación con JavaScript.
lang	Determina el idioma en el que está escrito el contenido del elemento.
spellcheck	Determina si el elemento debe ser analizado ortográfica y gramaticalmente por el navegador.
style	Permite asignar un estilo al componente.
tabindex	Determina el orden de selección de elementos cuando se pulsa el tabulador.
title	Permite especificar información extra al elemento.
translate	Determina si el elemento debe ser traducido o no.

En el siguiente código de ejemplo se pueden ver algunos de los atributos globales de HTML :

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h2 id="encabezado" class="rojo">Encabezado</h2>
    <p class="rojo" style="background-color: aquamarine;">
        Este párrafo será visible en el navegador
    </p>
    <p id="explicacion" hidden>
        Este párrafo no será visible en el navegador
    </p>
    <p contenteditable spellcheck="true" tabindex="2">
        Este parrafo tiene faltas de ortografia y es editable
    </p>
```

```
    <p lang="en" translate="no">
        This paragraph is written in English
    </p>
    <input type="text" tabindex="1">
    <input type="text" tabindex="3">
</body>
</html>
```

La representación generada por el navegador es la siguiente:



Algunas de las características que proporcionan los atributos globales no son visibles .Por ejemplo, el párrafo que tiene el atributo **hidden** no se muestra en el navegador. Tampoco la navegación con las pulsaciones de la tecla de tabulación: La primera pulsación posiciona del cursor en el primer elemento **<input>**; La segunda pulsación posiciona al cursor en el párrafo que tiene el atributo **contenteditable**; La tercera pulsación posiciona el cursor en el último elemento **<input>**. También hay atributos globales que tienen un efecto visible en la página: el color de fondo o el resultado de realizar la corrección ortográfica (depende del navegador). El resto de los atributos globales no tienen un efecto visible ni interno a *priori,* como el atributo **id** y el atributo **class**, que sirven para poder referenciar a los elementos desde las hojas de estilo CSS y desde programas escritos en Javascript.

Existen elementos como por ejemplo **<tt>** ó ** ** cuyo uso está desaconsejado en HTML 5, ya que están obsoletos. El mismo resultado que producen estos elementos se puede obtener utilizando estilos CSS y las recomendaciones hacerlo de esta manera.

2.3.1.- Elemento html

Es el elemento principal y la raíz del documento HTML. Todo el contenido del documento HTMLTodo el contenido del documento HTML agrupado de los elementos <head y <body> está incluido dentro de este elemento.

2.3.2.- Elemento head

Es la primera de las secciones de una página web no será directamente visible en la representación del documento ni en el navegador propiamente dicho (con alguna excepción), pero sí tendrá efecto directo y relevante en dicha representación.

El elemento más habitual que se incluye en esta sección es **<title>**. Contiene el título de la página, un texto que se muestra en la pestaña del navegador de web.

Además de mostrarse en la pestaña correspondiente del navegador, el título es el nombre que se asigna a una dirección web cuando se almacena como favorita en el navegador. El título también lo tiene en cuenta los buscadores de Internet para clasificar la página ,por lo que una correcta asignación de éste, hará que el posicionamiento sea más acorde al contenido de dicha página .

El elemento **<title>** no es obligatorio pero sí altamente recomendable, ya que en caso de no utilizarse el navegador, mostrará el nombre del fichero y esto, de cara al usuario final es una información irrelevante y probablemente confusa.

«La optimización del posicionamiento en buscadores, conocido como SEO (*Search Engine Optimization*) está formada por un conjunto de técnicas que pretenden mejorar la posición en la que aparecerá una página web al hacer una búsqueda relacionada en un buscador como Google. La etiqueta **<title>** es uno de los elementos que utilizan los buscadores para posicionar la página.»

Además del título, en esta sección se especifican los denominados **metadatos**. Un metadato es, en este contexto y utilizando una definición muy simplificada, un dato descriptivo que va a afectar a cómo se interpretan los datos contenidos en el documento.

Los metadatos se identifican con la etiqueta **<meta>** seguida de los atributos y valores que correspondientes a la información que se quiere proporcionar al navegador.

En la siguiente tabla se indican algunos de los metadatos más habituales:

Tabla 2.4. Los metadatos definen características generales del documento HTML

Metadato	Descripción
charset	Codificación del juego de caracteres. El valor estándar utilizado en HTML5 es UTF-8. Con anterioridad a esta versión de HTML el valor estándar era ISO-8859-1, por lo que es frecuente encontrar esta codificación en muchos documentos HTML.
http-equiv	Se utiliza para especificar una directiva. Todos los valores permitidos son nombres de encabezados HTTP.
name	Nombre del metadato. Se debe utilizar juntamente con el metadato **content.**
content	El valor asociado a atributos como **name** y **http-equiv.** Dependiendo del tipo de atributo, será un texto libre o una opción dentro de un conjunto.

Los valores más habituales del metadato «charset» son UFT-8 (codificación Unicode) e ISO-8859-1 (codificación del alfabeto latino).

```
<meta charset="UTF-8">
```

Los valores más habituales del metadato «charset» son UTF-8 (codificación Unicode) e ISO-8859-1 (codificación del analfabeto latino).

«La lista completa de juegos de caracteres que se pueden utilizar referentes al metadato «charset» se encuentra publicada en la web de la Internet Assigned Number Authority (**IANA**), que es la entidad que se encarga de supervisar asignaciones de recursos de Internet tales como las direcciones IP.

Se puede consultar la lista completa de juegos de caracteres aceptados en el enlace siguiente: http://www.iana.org/assignaments/character-sets/character-sets.xhtml.»

Por su parte, el metadato **<http-equiv>** tiene más alcance, por lo que la definición es sustancialmente más compleja. Mediante este metadato se pueden especificar aspectos tan diferentes como el tipo de caché que requiere la página, qué debe hacer el navegador ante la descarga de determinado documento, si el contenido está codificado con algún algoritmo de compresión, la fecha de publicación o cada cuánto tiempo debería recargar la página del navegador de manera automática.

En el siguiente listado se muestran algunos ejemplos de uso de este metadato junto con la descripción de su comportamiento:

- «**content-type**». Especifica el tipo de contenido y el juego de caracteres. En el ejemplo que se muestra a continuación es de tipo texto utilizando el lenguaje HTML y la codificación UTF-8.

  <meta http-equiv="**content-type**" content="text/html"; charset="UTF-8">

- «**cache-control**» Específica cómo ha de gestionar el navegador la página HTML desde el punto de vista de la caché. Se puede indicar una fecha de expiración, que sólo se almacena en memoria y no en soporte físico o desactivar por completo el almacenamiento en caché con el objetivo de que cada vez que se consulte la página se vuelva a descargar del servidor. En el siguiente ejemplo se está indicando al navegador que no almacena la página en caché.

  <meta http-equiv="**cache-control**" content="no-store">

- «**refresh**» Indica el navegador que pasando un determinado período de tiempo, refresque la página o, alternativamente cargue una página referenciada a través de una URL. En la etiqueta que se muestra como ejemplo a continuación se está indicando al navegador que pasados tres segundos, cargue la la página principal del sitio web de marca.

  <meta http-equiv="**refresh**" content="3; url=http://www.marca.com/">

Los posibles valores de los atributos **«name»** y **«content»** son muy diversos se hace referencia a una gran variedad de características que se pueden configurar para determinar el comportamiento de la página y del navegador A continuación se muestran algunos de ellos:

- «**robots**» Hace recomendaciones a los robots utilizados por los motores de búsqueda. Permite indicar, por ejemplo, que no se desea que se indexe una página o las páginas a las que dirigen los enlaces de la página actual. También se puede hacer saber al robot que se quiere que la página sea indexada, aunque esto lo suelen hacer por defecto. En el ejemplo siguiente se indica los potenciales robots que no indexe en la página:

  <meta name="**robots**" content="noindex" />

- «**keywords**» Indica palabras clave que servirán para que los indexadores de los motores de búsqueda sean capaces de clasificar de una manera más eficiente la página. El número de palabras clave puede ser variable y deberán hacer referencia al contenido de la página.

  <meta name="**keywords**" content="marca, html, futbol" />

- «**author**» Permite indicar el nombre del autor.

  <meta name="**author**" content="El nombre del autor" />

- «**copyright**» Permite indicar el propietario de los derechos de la página web.

  <meta name="**copyright**" content="Periódico de deportes" />

- «**viewport**» Permite determinar el ancho de la zona de representación (viewport) de la página, la escala inicial o las posibilidades de escalado que se le aplica a la página. Es fundamental para el diseño web adaptable (responsive design). En este caso, le dice al navegador que el ancho de la página debe ajustarse al ancho del dispositivo (width=device-width) y que el nivel de zoom inicial debe ser 1.0 (initial-scale=1.0).

  <meta name="**viewport**" content="width=device-width, initial-scale=1.0">

«URL son las siglas de *Uniform Resource Locator* (Localizador de recursos uniforme). Consiste en una secuencia de caracteres que identifica un recurso dentro de una red, normalmente Internet, y permite su localización. Una URL puede almacenar la dirección, por ejemplo, a una página HTML, una imagen, un vídeo o un documento XML.

Formato general de una URL: esquema://servidor/directorio/recurso.

Un ejemplo básico de URL: https://www.iesagora.es/incidencia »

2.3.3.- Elemento body

El elemento body delimita la sección que contiene los elementos que forman la parte visible de la página HTML Es por lo tanto la sección más importante ya que en ella se encuentra los contenidos que se van a presentar en el navegador.

El número de etiquetas que se puede incluir en esta sección es ingente y en esta unidad se encuentra se muestra una importante relación de estas.

Los elementos de la sección delimitada por el elemento <body> se clasifican en elementos **de bloque** y elementos **de línea**. Dichos tipos tienen las siguientes características.

Elementos **de bloque**.

- Ocupan todo el espacio de su elemento padre (su contenedor).

- Salvo excepciones, permiten contener a otros elementos, ya sean estos de bloque o de línea.

- Salvo excepciones, siempre comienzan y terminan con un salto de línea.

Elementos **de línea**.

- Ocupan lo que ocupa su contenido.

- Contienen datos u otros elementos de línea.

- Pueden comenzar en cualquier lugar de una línea y no generan salto de línea.

2.3.4.- Elementos de bloque

Los elementos de bloque forman un conjunto muy heterogéneo, ya que tienen un común características relacionadas con el espacio que ocupan y su distribución en la página. No obstante, es una de las categorías que se pueden utilizar para clasificar los elementos de HTML.

En esta sección Se muestra la relación de elementos de bloque, ordenados alfabéticamente. Se incluye su descripción, referencias a los atributos más significativos (exceptuando a los atributos globales) ocasionalmente código HTML de ejemplo junto con su representación visual y cuando proceda información complementaria que ayude a comprender el componente. La lista de atributos no es exhaustiva, ya que esta es muy extensa, por lo que se sugiere acudir a manuales de referencia o a documentación técnica para profundizar las características de cada componente.

«*MDN Web Docs* es una web de la Fundación Mozilla que contiene recursos para desarrolladores. Se puede profundizar en el conocimiento de HTML en general y de los elementos de bloque en particular a través de sus páginas.

Información relativa a los elementos en bloque:

https://developer.mozilla.org/es/docs/Web/HTML/Block-level_elements »

Los elementos de bloque permiten, entre otras cosas estructurar y organizar la página para facilitar su correcta interpretación por parte de los programas informáticos. Una posible estructura de página se muestra en la siguiente figura. En dicha estructura, del contenido del bloque **HEADER** podría contener, por ejemplo, los enlaces para la navegación general; El bloque **ASIDE**, los enlaces para navegación referente a la sección en la que se encuentre el usuario; el bloque **SECTION**, una sección contenedora de dos artículos incluidos en sendos bloques **ARTICLE**, y el bloque **FOOTER**, podría incluir referencias a la empresa propietaria de los derechos de la página, además de otra información.

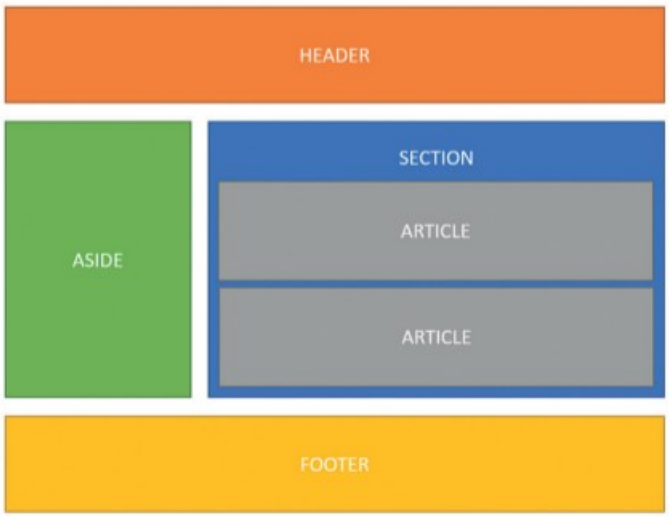


Figura 2.4. *La estructura de una página HTML la determina su autor.*

Cada uno de los contenedores, desde el más general hasta el más específico, es un elemento de bloque.

Esta estructura, como se ha comentado, es una referencia y admite múltiples variantes. Lo realmente importante es ubicar cada contenido en un bloque semántico correspondiente para facilitar el procesamiento automático del documento.

Se detallan a continuación los elementos que se engloban en esta categoría, excepto los elementos **<form>** y **<fieldset>** que se exponen en el apartado 2.5 y el elemento **<table>** que se expone en el apartado 2.4.

<address>

Descripción: representa información de contacto (dirección de correo electrónico, número de teléfono, etcétera) para el elemento **<article>** más cercano o para elemento **<body>** en el que se encuentra. El texto se muestra en cursiva.

Ejemplo:

```
<address>
Puede contactar con la editorial a través del sitio web <a href="https://www.
paraninfo.es/">www.paraninfo.es</a>
</address>
```

Puede contactar con la editorial a través del sitio web www.paraninfo.es

Figura 2.5. *El elemento <address> indica que su contenido es información de contacto.*

<article>

Descripción: Este contenedor agrupa información concebida como una unidad independiente. Algunos ejemplos serían una noticia de un periódico, un artículo de una revista, una entrada en un blog o un comentario en un foro. No genera ninguna representación gráfica específica.

<aside>

Descripción: Agrupa contenidos que están relacionados con el contenido principal del documento. Al ser un elemento de orientación puramente semántica, no tiene representación visual específica. No obstante, se representa habitualmente como una barra lateral en la que se muestran enlaces relacionados, publicidad o información complementaria al contenido principal del documento.

<blockquote>

Descripción: Contiene una cita.

Atributos significativos:

cite: Contiene la referencia a la fuente de la cita.

Ejemplo:

```
<blockquote cite='https://webfoundation.org/2017/03/web-turns-28-letter/'>
    <p>El modelo comercial actual de muchos sitios web ofrece contenido
gratuito a cambio de datos personales. Muchos de nosotros estamos de acuerdo
con esto, aunque a menudo al aceptar documentos de términos y condiciones
largos y confusos, pero fundamentalmente no nos importa que se recopile
cierta información a cambio de servicios gratuitos.</p>
</blockquote>
```

<details>

Descripción: Almacena un contenido textual que se hace visible cuando se «despliega» el componente.

Puede incluir como hijo el elemento **«summary»** para representar una etiqueta que identifique el contenido. De no usarse **«summary»** se mostrará una etiqueta genérica.

Se trata de una tecnología experimental, por lo que algunos navegadores pueden interpretar este elemento correctamente.

Ejemplo:

```
<details>
        <summary>Nota para el usuario</summary>
    El contenido de este manual está actualizado en relación a la versión
10.8 del producto.
</details>
```

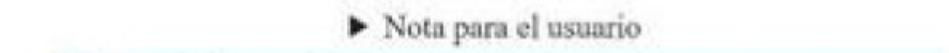


Figura 2.6. *El elemento <details> permite ocultar el contenido para que no ocupe espacio.*

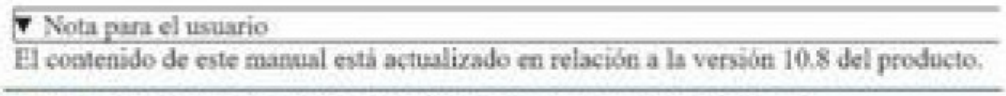


Figura 2.7. *Al interactuar con el elemento <details> se muestra el contenido oculto.*

<dialog>

Descripción: Muestra una caja de diálogo.

Atributos significativos:

open: Si está disponible, se muestra el diálogo ;en caso contrario, no se muestra.

Ejemplo:

```
<dialog open>
        <p>¡La compra se ha realizado correctamente!</p>
</dialog>
```

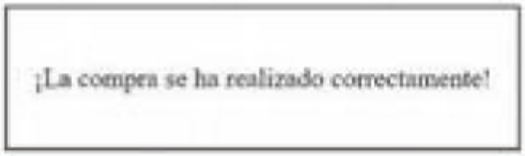


Figura 2.8. Las cajas de diálogo son útiles para captar la atención del usuario.

** **

Descripción: Permite crear divisiones y agrupar contenidos, sean éstos del tipo que sean. Se utiliza, junto con las hojas de estilo CSS, para maquetar y estructurar las páginas. No tiene una representación visual particular.

<dl>, <dt> y <dd>

Descripción: Usados conjuntamente permiten definir una lista de descripciones. **<dl>** marca la lista de definición, **<dt>** cada uno de los términos definidos y **<dd>** la definición de cada término.

Ejemplo:

```
<p>Géneros de videojuegos:</p>
<dl>
     <dt>Arcade</dt>
     <dd>Tienen como objetivo alcanzar una meta salvando obstáculo y/o
enemigos. Suelen ser muy dinámicos y tienen mecánicas sencillas.</dd>
     <dt>Estrategia</dt>
     <dd>Tienen como objetivo organizar una serie de recursos y maximizar su
rendimiento, ya sea este bélico, económico, social, etc.</dd>
     <dt>Disparos</dt>
     <dd>Tiene como objetivo alcanzar una meta o cumplir una misión en un
entorno hostil. También conocidos como <em>shooters</em></dd>
</dl>
```

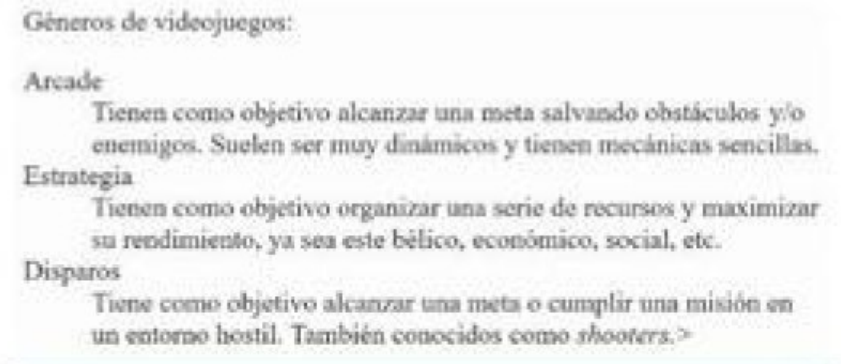


Figura 2.9. La lista de definición agrupa conceptos y descripciones.

Actividad propuesta 2.4

<figure> y <figcaption>

Descripción: Representa una ilustración, figura, diagrama, etcétera., y su leyenda o subtítulo (mediante el elemento **<figcaption>**).

Ejemplo:

```
<figure>
        <img src="logow3c.png" alt="Logotipo del W3C">
        <figcaption>El logotipo del World Wide Web Consortium (W3C)</
figcaption>
</figure>
```

El logotipo del World Wide Web Consortium (W3C)

Figura 2.10. Utilizando el elemento <figcaption> se mejora la accesibilidad de la página web.

«La accesibilidad de una página web es la capacidad que tiene para garantizar el acceso a la información contenida en ella a personas con algún tipo de discapacidad El W 3 C define una serie de pautas para garantizar que una página es accesible Puedes profundizar en la accesibilidad web a través de este enlace:

https://www.w3c.es/Traducciones/es/WAI/intro/accesibility »

<footer>

Descripción: Este contenedor representa un pie de página. Típicamente contiene información sobre la propiedad, autoría y los derechos del documento. Es un elemento semántico por lo que agrupa, pero no proporciona ninguna aportación visual.

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>

Descripción: Representan elementos de encabezado. Tienen distintos niveles de importancia. El elemento **<h1>** es el más importante y el elemento **<h6>**, el menos importante.

Cada elemento tiene una representación distinta en el navegador a mayor importancia mayor tamaño del texto que se puede replicar mediante hojas de estilo, pero esto no es una buena práctica.

Los agentes de usuario utilizan los elementos de encabezado para construir las tablas de contenido, lo que mejora la visibilidad de la página en los buscadores como Google.

Utilizar estos elementos mejora, además, el posicionamiento SEO, por lo que es muy importante utilizarlos de manera correcta.

Ejemplo:

```
<h1>Encabezado de nivel 1</h1>
<h2>Encabezado de nivel 2</h2>
<h3>Encabezado de nivel 3</h3>
<h4>Encabezado de nivel 4</h4>
<h5>Encabezado de nivel 5</h5>
<h6>Encabezado de nivel 6</h6>
```

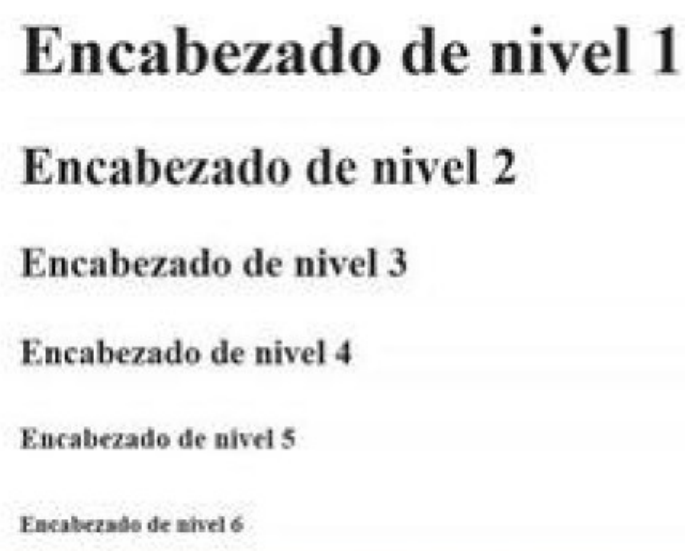


Figura 2.11. *Los elementos de encabezado son muy importantes para obtener un buen posicionamiento.*

«Generalmente se conoce como **agente de usuario** a la aplicación que está accediendo como cliente a un recurso proporcionado por un servidor web. Los navegadores web son ejemplos de agentes de usuario, pero también lo son los robots de búsqueda o de compra. En general, cualquier programa que acceda a contenido alojado en un servidor web de forma autónoma o dirigida por un usuario se puede considerar agente de usuario.»

<header>

Descripción: Contiene el encabezado de un documento o de una sección. En este elemento, de naturaleza semántica, se suelen ubicar los logotipos e imágenes corporativas, los formularios de búsqueda, la cesta de la compra de sitios web de venta online o enlaces de navegación de alto nivel. No tiene una representación visual específica.

<hr>

Descripción: Representa un cambio de tema entre párrafos mediante una línea de división. Aunque se representa como una línea, su principal objetivo es aportar información semántica. No puede ser contenedor.

**<li>, <ol>, **

Descripción: Representa una lista y sus elementos la etiqueta **<li>** identifica cada uno de los elementos de la lista. Por su parte, las etiquetas ** y ** identifican una lista ordenada (sus elementos aparecen numerados) y una lista no ordenada (sus elementos aparecen identificados con viñetas), respectivamente. Las listas se pueden anidar.

Ejemplo (lista ordenada):

```
<ol>
    <li>Subir al Kilimanjaro</li>
    <li>Visitar Nueva York</li>
    <li>Aprender a jugar al ajedrez</li>
    <li>Viajar en el tren Transiberiano </li>
</ol>
```

1. Subir al Kilimanjaro
2. Visitar Nueva York
3. Aprender a jugar al ajedrez
4. Viajar en el tren Transiberiano

Figura 2.13. *Las listas ordenadas se utilizan en relaciones de elementos cuyo orden es importante.*

Ejemplo (lista no ordenada):

```
<ul>
    <li>Lunes</li>
    <li>Martes</li>
    <li>Miércoles</li>
    <li>Jueves</li>
    <li>Viernes</li>
</ul>
```

- Lunes
- Martes
- Miércoles
- Jueves
- Viernes

Figura 2.14. *Las listas sin ordenar se utilizan en relaciones de elementos cuyo orden no es importante.*

Actividad propuesta 2.5

<main>

Descripción: Representa el contenido principal del documento. La motivación de esta etiqueta es principalmente semántica, ya que identifica de manera inequívoca el bloque de información principal de la página. Debe incluirse un único contenedor **<main>** dentro de un documento y éste no debe estar dentro de elementos **<article>, <aside>, <footer>, <header>** ni **<nav>,** por razones semánticas. No tiene representación específica.

<nav>

Descripción: Este contenedor está concebido para agregar enlace de navegación, ya sean en estos internos al documento o a otros documentos. Habitualmente se muestran las tablas de contenido, los índices, menús o sistemas de navegación alternativos. No tiene representación específica.

<p>

Descripción: Delimita un párrafo de texto. No tiene representación específica.

<pre>

Descripción: Representa un texto tal y como está en el documento HTML, respetando saltos de línea, tabulaciones y espacios múltiples. Utiliza una fuente de letra de ancho fijo.

Ejemplo:

```
<p>
    Este texto está dentro de un párrafo.
    Las tabulaciones, saltos de línea y los espacios      múltiples se ignoran.
</p>
<pre>
Este texto está dentro de un elemento pre.
       Los saltos de línea y los espacios      múltiples se respetan.
</pre>
```

Este texto está dentro de un párrafo. Las tabulaciones, saltos de línea y los espacios múltiples se ignoran.

```
Este texto está dentro de un elemento pre.
       los saltos de línea y los espacios      múltiples se respetan.
```

Figura 2.15. Solo se deben respetar espacios, tabulaciones y saltos de línea múltiple cuando sea necesario.

<section>

Descripción: Representa una sección genérica del documento. En ocasiones su uso se confunde con el de **<article>,** pero su diferencia es importante. Los elementos agrupados por **<article>** están relacionados entre sí y forman un bloque autocontenido e independiente de información. En cambio, **<section>** tiene como objetivo agrupar información relacionada y dependiente, aportando estructura semántica. No tiene una representación específica.

2.3.5.- Elementos de línea

AQUI Pag 48

2.3.6.- Contenido incrustado

AQUI Pag 63

2.4.- Tablas en HTML: estructura y elementos

AQUI Pag 72

2.4.1.- Elementos de las tablas

2.5.- Formularios en HTML: Estructura y elementos

Un formulario es un bloque de un documento HTML compuesto por un conjunto de elementos de entrada de información. Permite al usuario introducir datos con el objetivo de enviarlos a un servidor para realizar algún tipo de proceso con ellos.

En la web, los formularios son muy habituales; cualquier acceso mediante usuario y contraseña, la búsqueda de artículos en una tienda electrónica, la reserva de la habitación de un hotel o la consulta de la información meteorológica se realizan a través de un formulario. Es uno de los componentes más importantes en la interacción entre los usuarios y los sitios web.

Los formularios se delimitan por la etiqueta <form> y sólo tienen utilidad si contienen elementos de entrada de datos en su interior, ya que estos permitirán a los usuarios introducir la información que se debe enviar al servidor. En la construcción de formularios, por lo tanto, participan un importante número de elementos.

2.5.1.- Elementos de los formularios

Un formulario es un contenedor de elementos de entrada de datos. Estos elementos tienen como característica común que están orientados a permitir a los usuarios introducir información para, normalmente enviarla a un servidor y que sea debidamente tratada. No obstante, estos elementos se pueden utilizar fuera del

contexto de los formularios, ya sea para presentar información o para realizar algún tipo de programación mediante el uso de Javascript.

<form>

Descripción: Delimita una sección de una página HTML que contiene los componentes interactivos (formulario) que permiten al usuario introducir y enviar información a un servidor web.

Los formularios en HTML tienen un comportamiento específico y bien definido. Los componentes interactivos posibilitan introducir información y que éstas envíe al servidor, pero se han de cumplir algunas **reglas** sencillas.

1. Definir el destino (atributo **action)** y el modo de envío de los datos atributo (**method**).
2. Indicar de qué componentes queremos enviar los valores introducidos por el usuario (atributos **name**).
3. Especificar cuándo se quiere hacer el envío (elemento **input** del tipo **submit**).
4. Opcionalmente se pueden incluir reglas para validar los datos de entrada y éstas se validarán de forma automática antes del envío, cancelándose si dicha regla no se cumple.

Por tanto el **ciclo de funcionamiento** de un formulario es el siguiente:

1. El usuario introduce los datos en los campos de entrada correspondientes.
2. El usuario pulsa el botón de envío **submit**.
3. El navegador valida las reglas asignadas a los campos de entrada.
4. Si las reglas se cumplen, el navegador prepara los datos y los envía el destino indicado según el método de envío seleccionado.

En el siguiente formulario de ejemplo, se indica que el destinatario de los datos introducidos por el usuario en un programa escrito en lenguaje PHP ubicado en un servidor local, que el método de envío es GET y que los datos se van a enviar que se van a enviar son los que se introduzcan en los elementos **<input>** que tienen atributos **name**.

```
<form action="http://localhost/calculadoraimpuestos.php" method="GET">
    <input type="text" placeholder="Concepto" name="concepto">
    <input type="number" placeholder="Importe" name="importe">
    <input type="submit" value="Calcular">
</form>
```

En el navegador, el formulario anterior se muestra tal y cómo se puede observar en la Figura 2.51.

Figura 2.51. *El atributo placeholder permite mostrar un texto informativo en el campo input.*

En este mismo formulario se pueden incluir unas reglas que obliguen al usuario a introducir valores en los campos de entrada Para ello se utiliza el atributo **required**.

```
<form action="http://localhost/calculadoraimpuestos.php" method="GET">
    <input type="text" placeholder="Concepto" name="concepto" required>
    <input type="number" placeholder="Importe" name="importe" required>
    <input type="submit" value="Calcular">
</form>
```

Con esta modificación, al pulsar el botón de envío, el navegador verifica que los campos de entrada se han rellenado, cancelando el envío y mostrando un mensaje de error.

Figura 2.52. *Las validaciones de los campos se realizan al enviar los datos del formulario.*

En la declaración de un elemento **<form>** participan una serie de atributos de gran importancia, ya que determinan cómo se va a comportar el formulario. Gracias a estos atributos se pueden indicar el destino de los datos, el método de envío o la pestaña del navegador en la que se van a cargar En la siguiente tabla se muestran los atributos más importantes.

Tabla 2.5. Principales atributos del elemento <form>

Atributo	Valores más importantes	Descripción
action	URL de destino	Determina la URL (normalmente la dirección de un programa en un servidor web) que va a procesar el formulario.
method	GET, POST	Determina el método de envío. Con valor GET, los datos se envían como un anexo a la URL (separados de esta por el símbolo de cierre de interrogación «?»). Los datos serán visibles en el campo de dirección del navegador. Con POST, los datos se envían en el cuerpo de la petición y, por lo tanto, no son visibles en una navegación convencional.
target	_self (por defecto), _blank	Indica dónde mostrar el contenido devuelto por el servidor tras el envío del formulario. «_self» lo muestra en el mismo espacio (normalmente en la misma pestaña) en la que se mostró el formulario; «_blank» en un nuevo espacio (normalmente una nueva pestaña).
autocomplete	on, off	Indica si se deben autocompletar los campos con los valores introducidos en ejecuciones anteriores.

<<Los datos enviados con los métodos POST y GET siempre pueden verse desde el navegador. En el caso del método GET, estos se pueden observar a simple vista, ya que se muestran en la barra de direcciones del navegador. En el envío mediante el método POST, hay que acceder a las **<<Herramientas para desarrolladores>>**, incluidas en todos los navegadores, y en las que se pueden consultar los datos intercambiados entre el navegador y el servidor, entre ellos, los correspondientes a los formularios. Incluso los valores de las contraseñas introducidos en los elementos **<input>** de tipo **<password>** son visibles ya que este tipo de elemento de entrada de datos solo es una máscara gráfica y no realiza ningún tipo de cifrado ni codificación.>>

<input>

Descripción: Permite proporcionar un componente interactivo para que el usuario traduzca información para su envío a un servidor web. Es una etiqueta que contiene una enorme variedad de tipos de entrada, capaces de permitir introducir desde un texto simple hasta un color, pasando por fechas, contraseñas o valores numéricos dentro de un rango.

Los elementos **<input>** siempre recogen una cadena de caracteres con el objetivo de enviársela al servidor, independientemente del tipo de entrada que representen. El hecho de que haya tantos tipos permite proporcionar al navegador una funcionalidad más avanzada. Por una parte, ayuda en la introducción de datos, ya que los diferentes tipos de **<input>** utilizan máscaras y componentes gráficos de entrada que ayudan al usuario en la introducción de los datos. Por ejemplo, el tipo «**color»** proporciona un selector de color visual intuitivo .Sin ninguna duda, esta forma de elegir un color es mucho más sencilla que especificar el código RGB correspondiente De manera manual y sin ayuda.

Figura 2.53. Existen varios sistemas de color. RGB es uno de ellos.

<<RGB son las siglas en inglés de *Red Green and Blue* (rojo verde y azul). Es un modelo de color de los denominados **aditivos**, en el que cada color se representa especificando una intensidad para cada uno de los colores primarios.

Cada componente puede tomar un valor entre 0 y 255 (2ª combinaciones) y se asignan los valores de forma ordenada al rojo(R), verde(G) y azul(B).

Por ejemplo. la combinación de valores 255, 0, 0 representa el color rojo; la combinación 0, 255, 0 el verde; la combinación 0, 0, 255, el color azul; la combinación 0, 0, 0, el negro; la combinación 255, 255, 255 el blanco, y cualquier combinación con los tres valores iguales generará un torno de color gris.>>

Por otra parte, los componentes aportan seguridad, ya que contienen restricciones y validaciones para certificar que el tipo de dato que se está introduciendo es acorde al tipo de componente diseñado. Por ejemplo, el tipo **<<email>>** incluye una validación que comprueba antes de enviar los datos del formulario al servidor, que la dirección de correo electrónico introducida es correcta.

Figura 2.54. *En función del tipo asignado al elemento <input> se realizan unas validaciones concretas.*

Tabla 2.6. Principales valores del atributo «type» del elemento <input>

Tipo	Tipo de entrada	Representación visual
button	Botón de acción.	Activar
checkbox	Casilla para la selección de un elemento dentro de un conjunto de opciones.	
color	Color.	
date	Fecha.	03/09/2025
datetime-local	Fecha y hora.	24/12/2028 15:12
email	Dirección de correo electrónico.	info@paraninfo.es
file	Selección de un fichero del sistema de archivos.	Seleccionar archivo Lenguaje de marcas.docx
hidden	No es visible pero internamente existe a todos los efectos.	
image	Tiene el mismo comportamiento que el componente **<input>** de tipo **submit**, pero en lugar de texto muestra una imagen.	
month	Mes y año.	septiembre de 1971
number	Valor numérico.	35000
password	Contraseña. Visualmente sustituye los caracteres introducidos por asteriscos o puntos.	••••••
radio	Casilla para la selección de un único elemento dentro de un conjunto de opciones.	
range	Selección visual de un valor numérico dentro de un rango.	
reset	Restaura el estado inicial de los campos del formulario.	Restablecer

Tipo	Tipo de entrada	Representación visual
search	Texto para realizar una búsqueda. Tiene el mismo comportamiento que el elemento **<input>** de tipo **text**, pero permite establecer una diferenciación para realizar un tratamiento distinto.	
submit	Verifica y envía los datos del formulario al destino indicado en el atributo **action** de este.	Enviar
tel	Número de teléfono.	+34630001100558899
text	Texto libre.	
time	Hora.	18:15
url	URL.	http://www.paraninfo.es
week	Año y ordinal de la semana de dicho año.	Semana 30, 2032

Atributos comunes:

La lista de atributos del componente **<input>** es muy extensa, ya que cada uno de los tipos tiene unos atributos específicos que determinan qué valores se puede introducir y qué validaciones se deben realizar. Debido a la limitación propia de cualquier texto, la lista de atributos significativos de este componente se ha sustituido por la lista de atributos comunes a todos los tipos o a un gran número de ellos. Se recomienda profundizar en los atributos de cada uno de los tipos para poder conocer el detalle el componente **<input>**.

➔ **autocomplete**: Permite activar el autocompletado del componente.

➔ **autofocus**: El componente que tiene este atributo obtiene el foco cuando se carga la página, solo hay que utilizarlo, por lo tanto, en un elemento por página.

➔ **disabled**: Desactiva el componente.

➔ **form**: Contiene la referencia al **id** de un formulario, permitiendo que el componente pertenezca al formulario referenciado, aunque no sea un descendiente de éste.

➔ **list**: Permite asociar el componente con una lista de opciones definida en un elemento **<datalist>**. Válido para casi todos los tipos.

➔ **name**: Nombre del elemento de cara a su envío al servidor. Es importante recordar que si el elemento no tiene ese atributo, no será enviado, aunque esté contenido por el formulario.

➔ **placeholder**: Pese a que no es común en todos los tipos de elemento **<input>,** este atributo está presente bastante de ellos. Recibe como valor un texto y este se muestra en el espacio destirado de la entrada de datos. Desaparece cuando el usuario introduce el valor. Permite indicar al usuario qué tipo de dato espera el elemento **<input>**.

➔ **readonly**: Hace que el valor (atributo **value**) sea visible pero no editable. Válido para casi todos los tipos.

➔ **required**: Convierte el campo en obligatorio. De no rellenarse se mostrará un mensaje de error y la acción de envío de los datos del formulario al servidor no se realizará. Válido para casi todos los tipos.

➔ **type**: Determina la apariencia del componente, el tipo de dato que se va a introducir y la validación. No es obligatorio. Si se omite el tipo seleccionado por defecto es **text**.

➔ **value**: Valor por defecto del componente.

Actividad propuesta 2.16

<button>

Descripción: Permite crear un botón. Este botón no tiene una utilidad predefinida como, por ejemplo, los elementos **<input>** de tipo **submit** o **reset**, sino que deben programarse. No necesariamente han de ser utilizados dentro de un formulario.

Ejemplo:

```
<button>Enviar</button>
```

Figura 2.55. Se puede modificar el comportamiento de un botón utilizando JavaScript.

<select>

Descripción: Permite al usuario elegir una opción de un conjunto de ítems mostrado como una lista de opciones (puede mostrarse como lista desplegable). Las opciones establecen con el componente **<option>** y se pueden agrupar con el componente **<optgroup>**.

Atributos significativos:

multiple: Indica si admite selección múltiple.

size: Indica el número de líneas que se mostrarán simultáneamente.

Ejemplo 1:

En este primer ejemplo, al no modificarse el comportamiento por defecto del elemento **<select>**, se muestra una lista plegada con la primera de las opciones disponibles. Al pulsar sobre el componente se despliega la lista permitiendo seleccionar la opción deseada.

```
<select>
    <option>Baloncesto</option>
    <option>Fútbol</option>
    <option>Tenis</option>
    <option>Golf</option>
    <option>Esquí</option>
    <option>Remo</option>
</select>
```

Figura 2.56. Las listas desplegables ocupan poco espacio.

Figura 2.57. Las listas desplegables son más útiles con pocos elementos.

Ejemplo 2:

En este segundo ejemplo el atributo **size** tiene como valor «4», lo que provoca que aparezca una lista con cuatro elementos visibles. Por otra parte, el atributo **multiple** permite seleccionar múltiples opciones pulsando la tecla **Ctrl** y el botón izquierdo del ratón para hacer selecciones y deselecciones alternas, o pulsando la tecla **Shift** y el botón izquierdo del ratón para hacer selecciones y deselecciones por bloques.

```
<select multiple size="4">
    <option>Baloncesto</option>
    <option>Fútbol</option>
    <option>Tenis</option>
    <option>Golf</option>
    <option>Esquí</option>
    <option>Remo</option>
</select>
```

Figura 2.58. *Las listas desplegadas ocupan más espacio, pero muestran más información.*

<option>

Descripción: Este elemento define una opción seleccionable dentro de un elemento **<select>** o una sugerencia de un elemento **<datalist>**.

Atributos significativos:

disabled: Indica si la opción se encuentra deshabilitada.

label: Etiqueta la opción y se muestra sólo en el caso de que no haya contenido en el elemento.

selected: Indica que la opción debe estar seleccionada al cargar el documento.

value: Es el dato que se envía el servidor cuando se envía el formulario. No hay que olvidar que el contenido del elemento **<option>** tiene como único objetivo ser presentado en el navegador.

Ejemplo:

```
<select>
    <option value="opcion1">Seat</option>
    <option value="opcion2">Kia</option>
    <option value="opcion3" selected>Renault</option>
    <option value="opcion4" disabled>Ferrari</option>
    <option value="opcion5">Volvo</option>
</select>
```

Figura 2.59. *Los elementos <option> se pueden seleccionar por defecto y desactivar.*

<optgroup>

Descripción: Permite agrupar un conjunto de elementos **<option>** de un elemento **<select>**.

Atributos significativos:

disabled: Impide seleccionar la opción.

label: Determina el nombre del grupo de opciones.

Ejemplo:

```
<select>
    <optgroup label="Disponibles">
        <option value="opcion1">Seat</option>
        <option value="opcion2">Kia</option>
    </optgroup>
    <optgroup label="Sin stock">
        <option value="opcion4">Ferrari</option>
        <option value="opcion5">Volvo</option>
    </optgroup>
</select>
```

Figura 2.60. *Si el número de elementos es grande, resulta conveniente agruparlos.*

Actividad propuesta 2.17

<datalist>

Descripción: Define un conjunto de elementos **<option>** como una lista para su uso en otros elementos del formulario. Habitualmente en elementos **<input>** de tipo **text**, aparece como una lista de opciones al pulsar sobre el componente o genera una ayuda contextual filtrando las opciones según se escriben los caracteres en la caja de texto.

Ejemplo:

Es importante observar que el elemento **<datalist>** contiene un atributo **id** y que es referenciado por parte del atributo **list** del componente **<input>**. Este es el mecanismo para relacionar un componente de entrada con la lista de datos.

```
<input type="text" list="lenguajes" placeholder="Selecciona un lenguaje">
<datalist id="lenguajes">
    <option>Java</option>
    <option>C#</option>
    <option>Cobol</option>
    <option>Pascal</option>
    <option>Python</option>
</datalist>
```

Figura 2.61. Utilizando datalist se puede reutilizar parte del código HTML.

<fieldset>

Descripción: Permite agrupar un conjunto de campos de un formulario. La agrupación se realiza de forma visual mediante un marco, y de forma funcional, permitiendo realizar acciones sobre todo los elementos agrupados. Mediante el elemento **<legend>** se puede asignar un texto para que haga de título del marco.

Atributos significativos:

disabled: Permite deshabilitar todos los elementos agrupados.

form: El valor del atributo **id** del formulario al que está asociado.

name: Asigna un nombre al conjunto de campos.

Ejemplo:

```
<fieldset>
    <legend>Cliente particular:</legend>
    <p>Nombre: <input type="text" placeholder="Nombre"></p>
    <p>Dirección: <input type="text" placeholder="Dirección de envío"></p>
</fieldset>
```

```
<fieldset disabled>
<legend>Cliente empresa:</legend>
    <p>Nombre: <input type="text" placeholder="Nombre de la empresa"></p>
    <p>Dirección: <input type="text" placeholder="Dirección de envío"></p>
    <p>CIF: <input type="text" placeholder="Código de Identificación
Fiscal"></p>
</fieldset>
```

Figura 2.62. *Los elementos fieldset mejoran la legibilidad de los formularios.*

<legend>

Descripción: Permite asignar un título o leyenda a un conjunto de campos agrupados mediente un elemento **<fieldset>**.

<label>

Descripción: Identifica una etiqueta de un campo de entrada de datos del formulario. También permite poner el foco en el campo de entrada pulsando sobre el texto de la etiqueta. La asociación se puede realizar de dos maneras: conteniendo el campo de entrada entre las etiquetas de apertura y cierre del elemento **<label>** o utilizando el atributo **for** cuyo valor deberá ser el del atributo **id** del componente del formulario relacionado.

Atributos significativos:

for: Contiene la referencia al componente con el que está asociado el elemento **<label>**.

Ejemplo:

```
<p>
<label for="apodo">Apodo:</label>
<input type="text" id="apodo">
</p>
<p>
    <label>
        Dirección de correo:
        <input type="email">
    </label>
</p>
```

Figura 2.63. *El elemento label aporta mejoras en la usabilidad.*

<meter>

Descripción: Representa un valor dentro de un rango. Resulta útil para representar calificaciones, resultados en encuestas, grado de ocupación de un recurso, etcétera. Para para medir grados de avances más apropiado utilizar el componente **<progress>**.

En ausencia de los atributos **low**, **high** y **optimum,** todas las barras se representarán en color verde. Utilizando los atributos **low**, **high** y **optimum**, se establecerán tres segmentos: **min-low**, **low-high** y **high-max**. Si el valor de **optimum** está en el **último** segmento el color de la barra para un **value** que esté dentro de este se colorará en verde. Si el **value** está en el segmento anterior se colorará en naranja. Por **último**, si el **value** está en el primer segmento, se coloreará en rojo.

Atributos significativos:

value: Indica el valor actual.

min: Indica el valor mínimo.

max: Indica el valor máximo.

low: Indica el umbral por debajo del cual se considera que el valor es bajo.

high: Indica el umbral por encima del cual se considera que el valor es alto.

optimum: Indica el valor considerado óptimo.

Ejemplo 1:

```
<p>Avance del proceso: <meter min="0" max="100" value="20"></meter></p>
<p>Avance del proceso: <meter min="0" max="100" value="90"></meter></p>
```

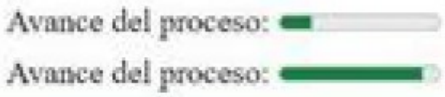


Figura 2.64. Es conveniente mantener informado a los usuarios del avance de los procesos.

Ejemplo 2:

```
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="4.9"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="5"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="6.9"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="7"></meter></p>
<p>Calificación: <meter min="0" max="10" low="5" high="7" optimum="8"
value="10"></meter></p>
```

Figura 2.65. Los codigos de colores facilitan la presentación de determinados datos.

<output>

Descripción: Delimita el resultado de una operación realizada normalmente por un programa.

Atributos significativos:

for: Contiene la relación separada por espacios de los identificadores de los elementos que participan en el cálculo. Es información semántica.

form: Incluye el identificador del formulario con el que está asociado.

name: Nombre con el que el valor del campo se envía al servidor.

Ejemplo:

```
<input type="number" id="operador1" value="8">
+
<input type="number" id="operador2" value="2">
=
<output for="operador1 operador2" name="resultado">10</output>
```

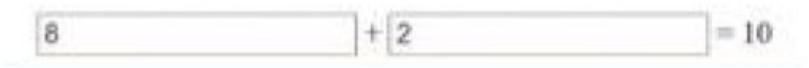


Figura 2.66. *No se deben utilizar elementos input para mostrar datos de salida.*

<progress>

Descripción: Muestra un indicador visual del grado de avance de un proceso. Si no se le asigna el atributo **value**, el indicador de avance aparece en movimiento de izquierda a derecha y de derecha a izquierda, informando de que el grado de avance del proceso es desconocido. No se debe confundir con el componente **<meter>**.

Atributos significativos:

value: Valor actual.

max: Valor máximo.

Ejemplo:

```
<progress></progress>
<progress max="10" value="3"></progress>
```

Figura 2.67. *Si es posible, es conveniente mostrar el grado de avance, ya que aporta más información.*

<textarea>

Descripción: Proporciona un área para la introducción de texto. La principal diferencia con elemento **<input>** de tipo **text** es que **<textarea>** permite introducir múltiples líneas. En la esquina inferior derecha dispone de un selector para cambiar el tamaño del componente con el ratón. Esta funcionalidad se puede desactivar mediante estilos CSS.

Atributos significativos:

cols: Determina el ancho del componente.

maxlength: Indica el número máximo de caracteres que admite.

minlength: Indica el número mínimo de caracteres que admite.

placeholder: Texto mostrado en el componente como ayuda para el usuario. Desaparece cuando se le asigna un valor al elemento.

readonly: Indica que el componente es de solo lectura.

required: Hace obligatorio introducir valor en el componente para cumplir con las reglas de validación.

rows: Número de líneas visibles. Si el texto excede del espacio disponible, se muestra una barra de desplazamiento vertical.

Spellcheck: Determina si el texto debe ser revisado ortográfica y gramaticalmente. Admite los valores true y false.

wrap: Indica el método de ajuste del texto para el envío al servidor. Admite los valores **soft** y **hard.** Visualmente ambos métodos se representan igual, rompiendo las líneas para que se vean por completo. Internamente el método **hard** envía dichos saltos de línea como parte del texto, mientras que el método **soft** envía el texto tal cual ha sido introducido.

Ejemplo:

```
<textarea name="hard" cols="40" rows="8" wrap="hard">

</textarea>
```

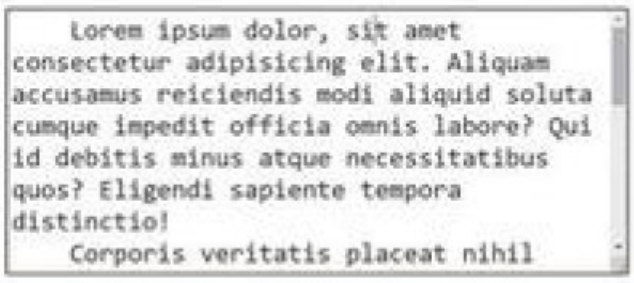


Figura 2.68. *El elemento textarea se debe utilizar para introducir bloques de texto.*

Actividad propuesta 2.18

2.5.2.- Validaciones

Antes de enviar un formulario al servidor es conveniente validar que los datos introducidos por el usuario cumplen con las restricciones que se hayan establecido. Por ejemplo, algunos campos de entrada serán obligatorios, los valores de determinados campos tendrán que estar dentro de los rangos definidos y la secuencia de caracteres introducida deberá seguir un patrón concreto.

Durante la definición de un formulario de entrada de datos, se especifican las condiciones que deben cumplir dichos datos, siendo ésta una tarea ligada íntimamente a la fase de diseño.

La validación de las reglas se realiza en una fase previa al envío del formulario mediante la pulsación de un elemento **<input>** de tipo **submit.** Alternativamente se pueden programar las reglas de validación en Javascript, pero esta opción siempre requiere más esfuerzo y deberá elegirse solo en caso de necesidad.

Todos los componentes de entrada de datos tienen atributos para definir reglas de validación. Algunos de los atributos referentes a la validación de formularios son los siguientes:

required: indica que el campo de entrada es obligatorio. Hay que tener presente que, en los campos de entrada de tipo texto, los espacios en blanco ya se consideran como valores válidos.

minlength y **maxlength**: determina la longitudes mínimas y máximas en campos de entrada de tipo texto.

min y **max** establecen los valores mínimos y máximos en campos de entrada numéricos.

type: en función del valor asignado a este parámetro se fijan los valores correctos de entrada. Por ejemplo, si el valor es **email**, se validará que el valor introducido es una dirección de correo electrónico.

pattern: comprueba que el valor introducido es acorde a una **expresión regular**.

Una expresión regular es un patrón formado por una cadena caracteres que define una regla. Es un mecanismo extremadamente flexible, pero que tiene un grado de complejidad relativamente alto. A continuación, un ejemplo para ilustrar en qué consisten.

Ejemplo:

Dado un campo de entrada **<input>** de tipo **text**, se desea que el valor introducido esté formado por una secuencia de caracteres formado por **un dígito numérico** seguido de **una letra minúscula** y **finalizado por el carácter #**. La expresión regular capaz de validar esta regla será la siguiente:

```
\d[a-z]#
```

Donde:

- \d indica que se debe introducir un dígito numérico.
- [a-z] señala que se debe introducir una letra minúscula entre la letra a y la letra z.
- # informa de que hay que introducir el símbolo de la almohadilla (#).

Valores válidos para esta expresión serán, por ejemplo: **1f#**, **8i#** y **0w#**.

En un formulario HTML, se puede incorporar esta validación en un campo **<input>** de tipo **text** de la siguiente manera:

```
<form action="">
    <input type="text" name="comando" pattern="\d[a-z]#">
    <input type="submit" value="Enviar">
</form>
```

«Una aplicación web consta de una parte denominada **cliente** (se ejecutan en el navegador) y otra parte denominada **servidor** (se ejecuta como indica su nombre en el ordenador servidor). Las validaciones se pueden realizar en ambas partes de la aplicación.

Es muy importante saber que las validaciones realizadas en el lado cliente son relativamente sencillas de **puentear** si se dispone de unos conocimientos técnicos intermedios sobre el desarrollo de aplicaciones web.

En caso de desarrollar una aplicación web con un nivel de fiabilidad medio o medio alto, es conveniente disponer de validación de datos en la parte ubicada en el servidor, ya que esta es mucho más difícil de evitar.»

Actividad resuelta 2.4 Formulario de registro

Actividad resuelta 2.4

Formulario de registro

Crear un documento HTML que contenga un formulario de registro a una tienda online. Deberán introducirse obligatoriamente los campos **nombre** y **dirección de correo electrónico**. El usuario podrá elegir si desea suscribirse a las notificaciones por correo electrónico o no. Los datos se deberán enviar usando el método POST a www.servidorlocal.edu/programa.php.

Solución

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Actividad Resuelta</title>
</head>
<body>
    <form method="POST" action="http://www.servidorlocal.edu/programa.php">
        <label for="nombre">Nombre:</label>
        <input type="text" name="nombre" id="nombre" required>
        <br><br>
        <label for="correo">Dirección de correo electrónico:</label>
        <input type="email" name="correo" id="correo" required>
        <br><br>
        <label for="recibir">Desea recibir notificaciones:</label>
        <input type="checkbox" name="recibir" id="recibir">
```

```
        <br><br>
        <input type="submit" value="Enviar">
    </form>
</body>
</html>
```

Nota sobre la solución

La separación entre las distintas partes del formulario se ha realizado utilizando elementos <br>. Existen mejores soluciones, pero estas pasan por la utilización de estilos CSS. Debido a que los estilos CSS se tratan en la Unidad 3, se ha preferido elegir una solución para el espaciado entre las partes del formulario menos correcta pero más comprensible en este momento del aprendizaje.

Actividad propuesta 2.19 Creación de un formulario de registro

2.6.- Multimedia en HTML

En HTML existe un breve pero potente conjunto de elementos multimedia. Gracias a ello s se puede componer una página que incluya imágenes estáticas, vídeos, mapas interactivos, música, sonidos, animaciones o videojuegos.

Los elementos multimedia han estado presentes en la web desde el principio mediante los elementos <img>. Durante esos primeros años las redes de comunicaciones habituales eran muy lentas y se debió optimizar su uso: se limitaba el tamaño de las imágenes al mínimo de resolución y tamaño necesario para su correcta visualización. Por suerte, las redes de comunicaciones han mejorado de forma drástica y todas estas restricciones prácticamente han desaparecido o se han relajado. En la actualidad es habitual ver contenido multimedia en alta resolución y emitido en directo *streaming* en casi cualquier hogar.

HTML 5 incrementó la capacidad de incluir multimedia en las páginas web mediante los componentes <audio< y <vídeo> proporcionando nuevas oportunidades para los diseñadores y programadores de páginas.

2.6.1.- Elementos multimedia

Con HTML 5 se introdujo un interesante conjunto de elementos para incluir contenidos multimedia a las páginas web, evitando utilizar soluciones no estandarizadas que pueden provocar problemas de compatibilidad. Estos elementos son muy completos y, a la vez, sencillos de utilizar. También disponen de propiedades para adaptar las los contenidos a los diferentes dispositivos, así como dotarlos de la capacidad de ser accesibles para personas con discapacidad.

<audio>

Descripción: Permite insertar recursos de sonido en la página. Es un componente muy sencillo de utilizar y está soportado por la gran mayoría de navegadores. Para establecer el recurso que reproducir se puede asignar mediante el atributo **src** o con el elemento **<source>**.

Admite proporcionar un contenedor alternativo por si el recurso no es compatible con el navegador.

Atributos significativos:

src: Determina la **URI** del recurso.

preload: Indica al navegador cómo se desea que se haga la precarga del recurso. Los valores posibles son **none** (no se precarga el recurso hasta que se vaya a utilizar minimizando el tráfico de datos) **metadata** (se recargan los los metadatos de recurso) o **auto** (se deja criterio del navegador).

autoplay: Indica al navegador que comience la reproducción automática del recurso tan pronto le sea posible. Debido a que la reproducción automática de sonido puede resultar una experiencia negativa para el usuario, la funcionalidad de reproducción automática está sujeta a una serie de condiciones adicionales.

loop: Indica que se desea realizar la reproducción en bucle.

muted: Si existe, silencia el componente.

controls: Si existe, se proporciona una interfaz para que el usuario pueda controlar la reproducción.

Ejemplo:

```
<audio src="rugido.ogg" controls loop>
Este contenido no es compatible con el navegador.
</audio>
```

Figura 2.69. *El reproductor de audio puede tener distinto aspecto en diferentes navegadores.*

Actividad propuesta 2.20 Reproductor de audio

<video>

Descripción: Permite insertar recursos de video en la página. Es un componente muy sencillo de utilizar y está soportado por la gran mayoría de navegadores. Para establecer el recurso que reproducir se puede asignar mediante el atributo **src** o con el elemento **<source>**. Permite proporcionar un contenido alternativo por si el recurso no es compatible con el navegador.

Atributos significativos:

src: Determina la **URI** del recurso.

poster: Establece la URI de una imagen para mostrar mientras recarga o inicia la reproducción del vídeo.

playsinline: Muestra el vídeo dentro de su área de reproducción en lugar de hacerlo a pantalla completa. Normalmente los navegadores tienen este comportamiento como predeterminado, por lo que, en esos casos, el atributo no tendrá efecto.

preload: Indica al navegador cómo se desea que se haga la precarga del recurso. Los valores posibles son **none** (no se precarga el recurso hasta que se vaya a utilizar minimizando el tráfico de datos) **metadata** (se recargan los los metadatos de recurso) o **auto** (se deja criterio del navegador).

autoplay: Indica al navegador que comience la reproducción automática del recurso tan pronto le sea posible. Debido a que la reproducción automática de sonido puede resultar una experiencia negativa para el usuario, la funcionalidad de reproducción automática está sujeta a una serie de condiciones adicionales.

loop: Indica que se desea realizar la reproducción en bucle.

muted: Si existe, silencia el componente.

controls: Si existe, se proporciona una interfaz para que el usuario pueda controlar la reproducción.

width y **height**: Determinan el ancho y alto del reproductor de vídeo.

Ejemplo:

```
<video src="trailerpelicula.avi" poster="posterpelicula.jpg" width="300"
height="300" controls>
Contenido no soportado
</video>
```

Figura 2.70. *El navegador y sus componentes determinan qué formatos de vídeo se pueden reproducir.*

<track>

Descripción: Permite incluir una pista con información sobre los elementos multimedia **<audio>** o **<vídeo>** en formato webVTT (extensión .vtt). Con este tipo de ficheros se puede almacenar información temporizada como, por ejemplo, los subtítulos.

Atributos significativos:

default: Determina el **<track>** que utilizar por defecto.

kind: Indica el tipo de pista. Admite las opciones **subtitles** (subtitulado), **captions** (transcripciones), **descriptions** (descripciones), **chapters** (capítulos) y **metadata** (información visible para el usuario y utilizada por programas).

label: Un título que describe la naturaleza de la pista.

src: Determina la **URI** del recurso.

srclang: Identifica el lenguaje del recurso.

Ejemplo:

```
<video width="800" controls>
    <source src="fuego.mp4" type="video/mp4">

    <source src="fuego.webm" type="video/webm">
    <track label="English" kind="subtitles" srclang="en" src="subtitle.vtt">
    <track label="Español" kind="subtitles" srclang="es" src="subtitulos.vtt"
default>
</video>
```

Figura 2.71. Los subtítulos posibilitan que personas con discapacidad auditiva accedan a la información.

<img>

Descripción: Permite Insertar una imagen en la página HTML a partir de un recurso externo referenciado mediante el atributo **src**.

El estándar no indica cuáles son los formatos que se deben aceptar por lo que la compatibilidad dependerá del navegador. Los formatos más habituales utilizados en web y soportados por los navegadores más populares son los siguientes:

Tabla 2.7. Formatos gráficos soportados por los navegadores web

Formato	Extensión	Tipo MIME
Animated Portable Network Graphics (APNG)	.apng	image/apng
Graphics Interchange Format (GIF)	.gif	image/gif
Joint Photographic Expert Group image (JPEG)	.jpg, .jpeg, .jpe, .jif, .jfif, .jfi	image/jpeg
Portable Network Graphics (PNG)	.png	image/png
Scalable Vector Graphics	.svg	image/svg+xml
Web Picture format (WebP)	.webp	image/webp

Atributos significativos:

alt: Contiene un texto con información equivalente a la proporcionada por la imagen. Se utiliza cuando la imagen no puede verse (usuario con discapacidad, formato soportado por el navegador).

src: Contiene la **URI** del recurso de imagen que se va a mostrar. Es un atributo obligatorio.

Srcset y sizes: Permiten asignar varias imágenes junto con indicaciones para ayudar al navegador a elegir la versión más adecuada.

usemap: Contiene la referencia a un mapa definido en un elemento **<map>**.

ismap: Si existe, indica que la imagen es parte de un mapa del lado del servidor.

width y height: Determinan el número de píxeles de ancho y alto, respectivamente, que ocupará la imagen. Si se indica uno de los dos atributos, el otro se calculará de forma automática para mantener la proporción original de la imagen. Si se indican ambos, la imagen se mostrará con el tamaño exacto indicado pero hay que tener en cuidado para no deformar la imagen.

decoding: Indica al navegador si la imagen debe ser decodificada de manera síncrona (valor **sync**) o asíncrona (valor **async**). Con el valor **auto** (por defecto), el navegador decide qué método utilizar.

loading: Permite indicar al navegador qué estrategia de carga de las imágenes utilizar. Con el valor **eager**, la carga se realiza inmediatamente en todos los casos; con valor **lazy,** la carga se difiere hasta que la sección de página visible en el navegador está próxima a la posición de la imagen.

<map>

Descripción: Dado un elemento **<img>**, permite definir áreas de navegación utilizando el elemento **<area>**. Estas áreas contendrán enlaces a otros recursos. Es decir, posibilita incrustar enlaces en regiones de una imagen para construir imágenes interactivas con mapas de navegación.

Atributos significativos:

name: Identifica al mapa para poder ser referenciado por el atributo **usemap** del elemento **<img>** que lo utilice.

<area>

Descripción: Define una región dentro de un elemento **<map>** permitiendo convertirla en un enlace a otro recurso. Admite varios tipos de figuras (rectángulos, círculos y polígonos) para delimitar el área. Contienen enlaces a otros recursos.

Atributos significativos:

alt: Versión textual del área.

shape: Tipo de figura que determina el área. Puede tener los valores **default** (toda el área del mapa), **rect** (una región rectangular) **circle** (una región circular) o **poly** (un polígono).

coords: Determina las coordenadas que forman el área. El formato depende del valor de **shape.**

href: La URI al recurso enlazado por el área.

target: Ubicación donde ha de abrirse el recurso enlazado. Al igual que el elemento **<a>**, admite los valores **_blank**, **_parent**, **_self** y **_top**.

download: Indica si el recurso debe abrirse o descargarse.

ping: Listado de URL a las que ha de conectarse el navegador cuando se selecciona el área.

rel: Información sobre el recurso enlazado.

<svg>

Descripción: Este elemento permite incrustar un gráfico vectorial en el documento HTML definiendo su aspecto mediante primitivas expresadas en XML. Su especificación completa está fuera del alcance de este módulo por lo que no se detalla su sintaxis ni estructura.

Ejemplo:

```
<svg xmlns="http://www.w3.org/2000/svg" width="150" height="100" viewBox="0 0
4 3">
  <rect width="4" height="1" y="0" fill="#00ab39" />
  <rect width="4" height="1" y="1" fill="#ffffff" />
  <rect width="4" height="1" y="2" fill="#000000" />
</svg>
```

Figura 2.73. Existen programas de diseño gráfico que generan código SVG.

Figura 2.72. Las imágenes interactivas pueden ser un recurso de navegación.

<canvas>

Descripción: Proporciona un lienzo sobre el que mostrar gráficos y animaciones mediante programas escritos en Javascript. Permite crear contenidos interactivos, anuncios animados o videojuegos. Este componente incorporado en HTML5 se gestiona de manera programática, por lo que la explicación detallada de su uso excede al objetivo de este módulo.

Atributos significativos:

width: Determina el ancho del componente.

height: Determina el alto del componente.

Ejemplo:

```
<canvas id="lienzo" width="500" height="500"></canvas>
```

2.7.- XHTML: La versión XML de HTML

Aqui Pag 105

2.8.- Herramientas de consulta, edición y validación de HTML

Aqui Pag 109

2.9.- Bibliografía y webgrafía

Bibliografía:

Editorial Paraninfo. Lenguajes de marcas y sistemas de gestión de la información. Fernando Paniagua Martín. 2025

Editorial Garceta. Lenguajes de marcas y sistemas de gestión de la información 2ª edición. Mª Isabel Jiménez Cumbreras. 2025

Webgrafía:

HTML Tutorial: https://www.w3schools.com/html/default.asp

CSS Tutorial: https://www.w3schools.com/css/default.asp

W3.CSS Tutorial: https://www.w3schools.com/w3css/default.asp

Lista de entidades HTML: https://html.spec.whatwg.org/multipage/named-characters.html#named-character-references

https://symbl.cc/es/html-entities/

Información relativa a los elementos en bloque de la fundación Mozilla:

https://developer.mozilla.org/es/docs/Web/HTML/Block-level_elements