
INTRODUCCIÓN A HTML5

2.1 INTRODUCCIÓN

El lenguaje HTML (HyperText Markup Language o lenguaje de marcado de hipertexto) es un lenguaje de marcado dedicado a la elaboración de páginas web. Fue definido por primera vez en 1991 y, en aquel entonces, se caracterizaba por tener algo más de una docena de etiquetas. Más tarde, en 1995 se publicó el primer estándar oficial de HTML al que denominaron HTML 2.0.

En 1997 entró en juego la W3C y desarrolló tres estándares más hasta llegar a lo que hoy conocemos como HTML5 en 2014.

Si bien HTML es un lenguaje formado por entidades que ayudan a estructurar y proporcionar significado a las diferentes partes del documento, cada una de estas entidades, usualmente denominadas elementos o etiquetas, están formadas por un contenido y cero, uno o varios atributos.

```
<p>Esto es un párrafo</p>  
<div class="layer">Esto es una capa</div>
```

Cada uno de los atributos tiene una función y puede estar o no asociado a un comportamiento o definición específica. Por ejemplo, el atributo ID habitualmente es utilizado para poder manipular el elemento a través de un nombre corto, sin embargo, también puede ser declarado para vincularse con otro elemento generando una entidad mayor, como es el caso del siguiente código.

```
<label for="nombre">Nombre</label>  
<input id="nombre" placeholder="Inserte el nombre completo" />
```

Ilustración 2.2. Etiquetado de un campo de formulario en HTML

El atributo FOR, utiliza el atributo ID para vincular el LABEL con el INPUT y generar un elemento combinado o pequeño componente.

Cabe destacar que, aunque puede haber etiquetas sin cierre, como es el caso del elemento INPUT, lo normal es que todas las etiquetas o marcas tengan un principio y un final, como es el caso de la etiqueta LABEL.

En lo referente a las novedades de HTML5, como muchos sabrán, una de las más significativas es el valor semántico. La semántica es una característica que dota a los documentos web de mayor significado porque, entre otras cosas, proporciona una mayor estructuración y ayuda a la compresión gracias a lo que se denomina identificador semántico.

El identificador semántico es un término que hace referencia a lo que contiene o representa la etiqueta, es decir, cada etiqueta o elemento tiene un nombre asociado que representa o indica su objetivo. Por ejemplo, en general, la etiqueta SECTION siempre contendrá un conjunto de elementos agrupados que tendrán o guardarán una relación.

2.2 ELEMENTOS BÁSICOS DE HTML

2.2.1 Definición del tipo de documento DTD (!DOCTYPE)

Cuando uno decide trabajar con HTML, lo primero que debe hacer es declarar el elemento !DOCTYPE. Este elemento tiene, como objetivo, informar al navegador del tipo de documento que se va a definir.

La Declaración del Tipo de Documento (DTD) puede cambiar, y de hecho cambia, para cada versión de HTML. La versión del lenguaje de marcado puede ser muy diferente según qué tipo se utilice, y puede tener más o menos restricciones en función del modo y versión. Sin ir más lejos, el tipo de documento que se debe definir para indicar que es un documento XHTML es muy distinto al que se debe usar para indicar que es HTML5 o SVG.

A continuación, se muestran los principales DTD para documentos de HTML, SVG y MathML.

2.2.1.1 DTDS DE HTML

HTML5

```
<!DOCTYPE html>
```

2.2.1.2 DTDS DE MATHML

MathML 2.0

```
<!DOCTYPE math PUBLIC "-//W3C//DTD MathML 2.0//EN"  
"http://www.w3.org/TR/MathML2/dtd/mathml2.dtd">
```

2.2.1.3 DTDS DE SVG

SVG 1.1 Full

```
<!DOCTYPE svg PUBLIC  
"//W3C//DTD SVG 1.1//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
```

SVG 1.1 Básico

```
<!DOCTYPE svg PUBLIC  
"//W3C//DTD SVG 1.1 Basic//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-basic.dtd">
```

SVG 1.1 Reducido

```
<!DOCTYPE svg PUBLIC  
"//W3C//DTD SVG 1.1 Tiny//EN"  
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11-tiny.dtd">
```

2.2.1.4 DTDS DE XHTML

XHTML1.1

```
<!DOCTYPE html PUBLIC  
"//W3C//DTD XHTML 1.1//EN"  
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

2.2.2 Etiqueta html

La etiqueta HTML es el elemento que representa la raíz o base de un documento HTML y supone el cierre automático del resto de los elementos declarados posteriormente a él.

La etiqueta HTML admite varios atributos, la mayoría en desuso. El único que sigue estando vigente es LANG y es el encargado de definir el lenguaje del documento.

Es un atributo muy útil cuando se dispone de documentos en distintos idiomas y para aquellos usuarios que dependen de herramientas de asistencia como lectores de pantalla.

```
<html lang="es">...</html>
```

2.2.3 Etiqueta head

La etiqueta HEAD es el elemento o la estructura que proporciona información general acerca del documento. Esta información general viene definida a modo de metadatos, o lo que es lo mismo, datos que informan sobre los datos y, pueden ser de muy diferente índole. Esto es, el tipo de codificación, el título del documento, las palabras clave que lo describen, la descripción sobre lo que contiene, el autor del documento, etcétera.

No obstante, lo que más abunda dentro de esta estructura suele ser elementos LINK o STYLE, los cuales recogen todas las reglas CSS aplicables en el documento.

```
<head>
<!-- Información del documento -->
</head>
```

NOTA

Aunque la etiqueta SCRIPT puede estar definida dentro del elemento HEAD, lo mejor es que esté al final de la etiqueta BODY para evitar bloqueos o retrasos en la muestra del primer renderizado.

2.2.4 Etiqueta body

La etiqueta BODY es el elemento o la estructura que define todo el contenido útil del documento. Aquí es dónde se definirán todos los textos, capas, botones, controles de entrada y salida, etcétera para que los usuarios puedan utilizarlo o consultarlo.

Al final de esta estructura, habitualmente, suele contener uno o varios elementos SCRIPT que todas las funcionalidades que se ejecutan en el navegador, como validaciones o animaciones.

```
<body>
<!-- Contenido del cuerpo de la página -->
</body>
```

2.2.5 Comentarios

Los comentarios en HTML se establecen a través de las marcas <!-- y -->. Estas etiquetas o marcas indican al navegador que la información contenida no debe ser interpretada y, por tanto, tampoco renderizada. Un ejemplo podría ser:

```
<!-- Esto es un comentario de HTML -->
```


2.3 INFORMACIÓN DEL DOCUMENTO

Los elementos que proporcionan información sobre el documento son las etiquetas `BASE`, `LINK`, `META`, `SCRIPT`, `STYLE` y `TITLE`.

2.3.1 Elemento base

El elemento `BASE` especifica la base de direccionamiento predeterminada para los elementos que utilicen un direccionamiento relativo, es decir, las direcciones no tienen como primer carácter el símbolo de barra inclinada (/) y no empiezan por un identificador de dominio.

Debe estar dentro del elemento `HEAD` y es importante destacar que, si se declaran varios elementos `BASE`, la última declaración anulará todas las anteriores.

```
<head>
<base href="https://www.ejemplo.com/" target="_blank" />
</head>
```

2.3.2 Elemento link

El elemento `LINK` especifica un enlace hacia una hoja de estilos o archivo externo.

```
<link rel="stylesheet" type="text/css" href="custom.css" />
```

El único atributo obligatorio del elemento `LINK` es `REL`, que es quién define la relación que existe entre el documento actual y el enlazado, es decir, si es una hoja de estilos, un icono, un archivo de ayuda, un documento de alternativa, ...) y su valor más recurrente es `STYLESHEET`.

No obstante, aunque el establecimiento de este atributo es importante y ayuda a la semántica web, también puede influir en el rendimiento de forma notable haciendo que los recursos se carguen de maneras diferentes.

Por esta razón, es importante que se conozca bien la aplicación que se está creando y optimizar los recursos, porque un mal uso del atributo `REL` puede influir negativamente en el rendimiento de la página.

A continuación, se muestran algunos de los valores de `REL` que afectan al rendimiento:

Valor	Descripción y ejemplo
length	<p>La cláusula DNS-PREFETCH indica al agente de usuario que se resuelva el DNS lo antes posible, es decir, que se resuelva el dominio del servidor, pero no realice ninguna descarga.</p> <p>Es especialmente útil cuando se desean cargar archivos desde fuentes externas como pueda ser una fuente vectorial de Google Fonts, un script de un CDN o un JSON desde una API.</p> <pre><link rel="dns-prefetch" href="//fonts.googleapis.com"/></pre>
prefetch / preload	<p>Las cláusulas PREFETCH y PRELOAD indican al agente de usuario que descargue y almacene en caché un recurso determinado, como pueda ser una hoja de estilos o un script.</p> <p>Mientras que la cláusula es PREFETCH provocará que la descarga se realice con prioridad baja, es decir, sin afectar a los recursos más importantes o prioritarios, la cláusula es PRELOAD provocará que la descarga se realice a la mayor brevedad posible, pudiendo afectar a los recursos más importantes o prioritarios.</p> <p>Ambas cláusulas se alimentan del atributo AS, que proporciona una pista de la prioridad del recurso al agente de usuario. Sus principales valores son STYLE, SCRIPT, FONT e IMAGE, aunque hay más.</p> <pre><link rel="preload" href="/js/scripts.js" as="script"></pre>
preconnect	<p>La cláusula PRECONNECT indica al agente de usuario que resuelva el DNS y realice la preconexión con los protocolos TCP y TLS, si procede.</p> <p>Al igual que DNS-PREFETCH, es especialmente útil cuando se desean cargar archivos desde fuentes externas como pueda ser una fuente vectorial de Google Fonts, un script de un CDN o un JSON desde una API. Sin embargo, su uso excesivo puede provocar pérdidas sustanciales en el rendimiento global, por lo que no se recomienda usarlo más de 4 o 6 veces por página.</p> <pre><link rel="preconnect" href="//islavisual.com/"></pre>
prerender	<p>La cláusula PRERENDER indica al agente de usuario que se renderice el recurso en segundo plano. Esto puede ser una buena idea cuándo se está seguro de que, el usuario, realizará alguna acción que requiera esta precarga.</p> <p>Su uso puede aumentar el rendimiento hasta un 70%, no obstante, este tipo de procesamiento es muy costoso, tanto a nivel de memoria, como a nivel de tráfico.</p> <p>Cabe destacar que, a febrero de 2023, esta cláusula no está del todo soportada por Firefox ni Safari, por lo que su uso no está recomendado.</p> <pre><link rel="preconnect" href="//islavisual.com/"></pre>

Con respecto a su posible personalización, permite, entre otras cosas, establecer el idioma en el que está escrito a través del atributo HREFLANG y el dispositivo o medio para el que está optimizado mediante el atributo MEDIA, que habitualmente es ALL, SCREEN o PRINT.

2.3.3 Elemento meta

Los metadatos se pueden definir como datos acerca de los datos. Es como una información que nos proporciona datos clave sobre el contenido que va a ser representado.

Por tanto, podríamos decir que los metadatos proporcionan información a los robots y usuarios que lo necesiten sobre el documento actual. No obstante, esta información puede ser de muy diversa índole, desde información referente a la apariencia, hasta datos que pueden ser utilizados por los agentes de usuario.

Dicho esto, el elemento META especifica una información sobre los datos que se encuentran dentro del actual documento HTML. Un ejemplo podría ser:

```
<meta name="keywords" content="HTML5, HTML, XHTML">
```

El elemento META se alimenta de varios atributos que permiten personalizar, entre otras cosas, la codificación de caracteres, respuestas al documento o el esquema que se debe emplear, lo cual pasamos a ver a continuación.

2.3.3.1 ATRIBUTOS DEL ELEMENTO META

2.3.3.1.1 Atributo name

El atributo NAME especifica el nombre clave que va a asignarse al metadato y asociarse a su par nombre-valor determinado por el atributo CONTENT.

En general, podemos decir que, el atributo NAME permite definir el tipo de metadato que se va a especificar. Esto es, por ejemplo, una descripción, un autor, las palabras clave (KEYWORDS) que clasifican el documento o, el tamaño y escalamiento de la ventana gráfica.

```
<meta name="author" content="Pablo E. Fernández Casado">
```

2.3.3.1.2 El atributo content

El atributo CONTENT especifica el valor de los metadatos referidos a la página o a directivas pragma.

El atributo de CONTENT sólo debe establecerse si se define el atributo NAME o el atributo HTTP-EQUIV por lo que, si el metadato que se está definiendo no contiene ninguno de estos atributos, se debe obviar su declaración para evitar errores de procesamiento, accesibilidad y/o usabilidad web.

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


2.3.3.1.3 El atributo charset

El atributo CHARSET se utiliza para especificar la codificación de caracteres en la que viene definido el documento. Entre sus valores más recurrentes podemos encontrar el valor UTF-8 y el valor ISO-8859-1.

El atributo CHARSET sólo puede ser interpretado una única vez por documento, es decir, que debe ser declarado una única vez en la página. Además, debe estar declarado al principio del documento, entre los primeros 512 bytes, dentro de la etiqueta HEAD porque, en caso contrario, puede ser ignorado.

Cabe destacar que, aunque existen multitud de posibles codificaciones, ningún agente de usuario las soporta todas, por lo que es importante que uno se informe antes de establecer una codificación de caracteres específica.

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

2.3.3.1.4 El atributo http-equiv

El atributo HTTP-EQUIV se dice que es una directiva pragma y vienen a ser simulaciones de encabezados de respuesta HTTP. En otras palabras, se utiliza para comunicarse con los servidores y adaptar sus respuestas al documento.

Este atributo era utilizado antes de la versión 5 de HTML para especificar la codificación de caracteres, pero ahora ya no está permitido. A continuación, se muestra una lista con los valores actuales más representativos que admite este atributo.

- **CONTENT-SECURITY-POLICY:** especifica una política de contenido que sirve para prevenir y disminuir algunos tipos de ataque como la inyección de datos o XSS (Cross Site Scripting) o para definir algún comportamiento que, se desea, se cumpla. Por ejemplo, si se desea que todo el contenido provenga del mismo origen, se puede establecer esta directiva a DEFAULT-SRC 'SELF'. Si, además, se desea que incluya todos sus subdominios, podría definirse la directiva a DEFAULT-SRC 'SELF' *.COMPONENTS.COM. (Para más información puede visitarse la URL <https://developer.mozilla.org/es/docs/Web/HTTP/CSP>).
- **DEFAULT-STYLE:** especifica la hoja de estilos por defecto o preferida. El valor de CONTENT debe ser exactamente el mismo que el definido en el atributo TITLE del elemento LINK o elemento STYLE.
- **REFRESH:** especifica un intervalo de tiempo, tras el cual, el documento se actualiza automáticamente.

```
<meta http-equiv="refresh" content="60">
```

2.3.3.1.5 El atributo scheme

El atributo SCHEME *no está soportado por HTML5*, no obstante, antes se utilizaba para especificar el esquema que se debía emplear para interpretar el valor de una propiedad.

En otras palabras, especificaba el esquema de formato o URI que se debía emplear para interpretar el valor del atributo CONTENT.

```
<meta name="date" content="01-01-2020" scheme="DD-MM-YYYY">
<meta name="identifier" content="0-2345-6634-6" scheme="ISBN">
```

2.3.4 Elemento title

El elemento TITLE especifica el nombre del recurso o documento para darlo a conocer.

```
<title>Curso de creación de páginas web</title>
```

En lo referente a los posibles valores admitidos, puede ser cualquier valor de texto que respete las normas de ortografía y gramática. Esto es, no se debe capitalizar la descripción del título, a no ser que sea un nombre propio, y se deben respetar los signos de puntuación.

2.3.5 Elemento style

Permite definir información de estilo a través de selectores y reglas CSS. Un ejemplo podría ser:

```
<style media="screen" type="text/css">
html{
font-family: Arial, sans-serif;
font-size: 14px;
font-style: normal;
}
</style>
```

El elemento STYLE tiene dos atributos, el tributo MEDIA y el atributo TYPE. El atributo MEDIA, permite especificar el dispositivo o medio para el que está optimizado el recurso y el atributo TYPE, especifica el tipo de medio o dispositivo, sin embargo, actualmente sólo admite el valor expuesto en el ejemplo, es decir, TEXT/CSS.

2.3.6 Elemento script

HTML puede ser de gran ayuda cuando se trata de describir el contenido, sin embargo, la inmensa mayoría de las veces se suele requerir de una funcionalidad que no nos proporciona el lenguaje. Sirva como ejemplo que, si lo que se desea es saber si el valor de un campo de entrada es válido, se debe recurrir a algún fragmento de código externo en lenguaje script para poder realizar las verificaciones pertinentes.

Dicho esto, el elemento SCRIPT permite insertar un código, o fragmento de código, ejecutable dentro de un documento HTML.

```
<script src="./validarNIF.js"></script>
```

2.3.7 Principales metadatos a definir

En la actualidad, existen gran cantidad de metadatos posibles y cualquiera de ellos puede definirse a través de los atributos NAME, HTTP-EQUIV, CHARSET, SCHEME e SCHEME. Por esta razón sólo presentaremos los más relevantes a nivel introductorio.

Entre los principales metadatos que debe haber en una página web deben estar la codificación de caracteres de la página (generalmente “UTF-8”), el título del documento, la definición del área útil o ventana de visualización y la inclusión de una hoja de estilos y un archivo con los scripts utilizados.

Por tanto, basándonos en esta aseveración anterior, lo único que nos queda es explicar lo que es el área útil o ventana de visualización y para qué sirve.

Viewport (definición del área útil o ventana de visualización)

El término VIEWPORT significa ventana y, en el contexto páginas web, se refiere al área útil de la pantalla dónde se visualizará el contenido.

En líneas generales, se puede afirmar que el tamaño de una página o documento renderizado no se corresponde con el tamaño del área visible de la pantalla. Por este motivo, cuando no se declara esta directiva, los contenidos pueden mostrarse con las barras de desplazamiento horizontal y vertical.

Pensemos, por ejemplo, en cómo puede verse una página, que está diseñada para ser visualizada en pantallas grandes, en un dispositivo móvil. Si esto sucede e intentamos visualizar un contenido pensado para ser mostrado en resoluciones superiores a 1280 píxeles, en un dispositivo que sólo dispone de 360 píxeles, el resultado será que, el usuario, tendrá que desplazarse tanto vertical, como horizontalmente, para acceder a todo el contenido.

Cierto es que este problema, en parte, podría evitarse no utilizando elementos con ancho fijo ni medidas absolutas como son los píxeles, pero, aun así, con todo y con ello, el resultado no sería completamente usable y accesible.

Pues bien, el VIEWPORT o área útil dónde se mostrará el documento, cambia en función de si estamos en un dispositivo de escritorio o en un dispositivo móvil. Si el navegador o herramienta de asistencia se ejecutan en un dispositivo de escritorio, el tamaño de la pantalla coincidirá con el tamaño del área útil para renderizar el documento, no así, cuando se ejecuta en un dispositivo móvil.

Cuando el navegador o herramienta de asistencia se ejecutan en un dispositivo móvil, el VIEWPORT no se corresponde con el tamaño real de la pantalla, sino con el espacio que la aplicación de software está emulando. Por ejemplo, en un dispositivo de Apple, aunque el ancho de la pantalla sea de 320 píxeles, en realidad se pueden estar emulando 980, porque el sistema realiza una serie de extrapolaciones para que se ajuste al espacio visible y se muestre el contenido de la mejor forma posible.

Todo esto, entre otras razones, es porque los dispositivos móviles poseen una densidad en píxeles muy diferente a las pantallas de escritorio. Mientras que en las pantallas de sistemas de escritorio la proporción de densidad en píxeles es de 1:1, en los dispositivos móviles puede llegar a ser muy superior.

NOTA

La densidad de píxeles es un valor que se calcula a partir de los valores de resolución y tamaño de la pantalla. Este valor especifica el número de píxeles que es posible mostrarse en una pulgada (2,54 cm).

Dicho esto, la directiva VIEWPORT especifica cuál debe ser el tamaño del área útil de pantalla y los valores posibles de escalado. Estos valores de escalado indican el nivel de escalado inicial (zoom) y si se puede o no ampliar o reducir.

```
<meta name="viewport" content="width=device-width, initial-scale=1 />
```

Entre los posibles valores que admite esta directiva tenemos:

Valor	Descripción y ejemplo
height	Especifica el alto, en píxeles, de la ventana gráfica. Permite definir cualquier valor entero positivo o la palabra clave DEVICE-HEIGHT, que indica que se utilice el 100% de alto de la pantalla.
initial-scale	Especifica la proporción (a escala) que existe entre el ancho o alto del dispositivo y el tamaño de la ventana gráfica. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.
maximum-scale	Especifica el nivel de escalado máximo que se puede utilizar, teniendo que ser, este, igual o superior al determinado por la propiedad MINIMUM-SCALE. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.
minimum-scale	Especifica el nivel de escalado mínimo que se puede utilizar, teniendo que ser, este, igual o menor al determinado por la propiedad MAXIMUM-SCALE. Permite definir cualquier valor, entero o decimal, comprendido entre 0 y 10.
user-scalable	Especifica si el usuario puede o no realizar cambios en el nivel de escalado (puede hacer zoom) en la página. Sus posibles valores son YES y NO. Por defecto el valor es YES.
width	Especifica el ancho, en píxeles, de la ventana gráfica. Permite definir cualquier valor entero positivo o la palabra clave DEVICE-WIDTH, que indica que se utilice el 100% del ancho de la pantalla.

Cabe destacar que, si establecemos esta directiva en un documento o página web que no presenta un diseño adaptativo y lo ejecutamos en un dispositivo móvil, el efecto que se producirá será que todo se hará mucho más pequeño para que entre en el área

útil o ventana gráfica. Por esta razón, esta directiva debe estar declarada y procesada en combinación con las consultas de medios de CSS (Media Queries) de manera eficiente, para que el contenido sea usable, legible y accesible.

2.4 ATRIBUTOS GLOBALES

2.4.1 Atributo accesskey

El atributo ACCESSKEY especifica un atajo de teclado (o combinación de teclas) que el usuario puede pulsar para interactuar con el elemento.

```
<button accesskey="h">Ayuda</button>
```

Como se puede observar en la ilustración anterior, la tecla de activación es la letra H, no obstante, dependiendo del navegador y sistema operativo, se activará de una u otra manera.

Navegador	Windows	Linux	iOS
Firefox	Alt+Shift+Tecla	Alt+Shift+Tecla	Command+Alt+Tecla
Edge	Alt+Tecla	n/a	
Internet Explorer	Alt+Tecla	n/a	
Chrome	Alt+Tecla	Alt+Tecla	Command+Alt+Tecla
Safari	Alt+Tecla	N/A	Command+Alt+Tecla
Opera 15+	Alt+Tecla	Alt+Tecla	Command+Alt+Tecla

Es por esta razón que la utilización del atributo ACCESSKEY puede afectar de forma considerable a la accesibilidad web ya que, un atajo de teclado programado con este atributo, puede entrar en conflicto con la herramienta de asistencia que esté utilizando el usuario e, incluso, puede pasar que el usuario presente una discapacidad cognitiva y el atajo de teclado le resulte confuso.

2.4.2 Atributo autocapitalize

El atributo AUTOCAPITALIZE especifica si el contenido que se va a representar debe capitalizarse de manera automática o no. Sus posibles valores son **OFF / NONE**, que indican al navegador que no se transforme el contenido, **ON / SENTENCES**, que indican al navegador que el contenido se transforme como si fuese una oración, es decir, el primer carácter del párrafo o elemento, **WORDS**, que le indica al navegador que se capitalice cada una de las palabras del contenido y **CHARACTERS**, que le indica al navegador que se transforme todo a mayúsculas.

```
<label>
Nombre:
<input type="text" name="name" autocapitalize="words">
</label>
```

i NOTA

Cabe destacar que esta funcionalidad no está soportada por Firefox Opera ni Safari. No obstante, es posible reproducir todas estas funcionalidades a través de las propiedades FONT-VARIANT y TEXT-TRANSFORM de CSS.

2.4.3 Atributo autofocus

El atributo AUTOFOCUS especifica si el elemento debe enfocarse automáticamente cuando finaliza el proceso de carga de la página o cuando se muestra un elemento de diálogo.

```
<label for="name">Nombre:</label>
<input type="text" id="name" name="name" autofocus>
```

Sin embargo, su uso no es recomendable porque puede confundir a las personas con problemas de visión que usan tecnología de lectura de pantalla y a las personas con problemas cognitivos. Esto es porque las acciones automáticas imprevistas, como llevar el foco automáticamente sin previo aviso a un control de formulario puede sorprender y frustrar a los usuarios.

2.4.4 Atributo class

El atributo CLASS especifica una definición de clase a la etiqueta o elemento. A diferencia del atributo ID, el valor de este atributo puede repetirse en cualquier parte del documento.

```
<p class="toLeft">
Este párrafo tiene una clase que hace que se alinee a la izquierda
</p>

<p class="toRight">
Este párrafo tiene una clase que hace que se alinee a la derecha
</p>
```

2.4.5 Atributo contenteditable

El atributo CONTENTEDITABLE especifica al navegador que el elemento es editable, aunque originalmente no lo fuese. Es decir, permite que el elemento pueda ser modificado en lo referente a su contenido, formato y estilos y pueda ser manipulado a través de los eventos asociados a la edición, como puedan ser el evento FOCUS o el evento KEYPRESS.

```
<p contentEditable="true">
Este párrafo es modificable,
<span>y esta etiqueta también.</span>
</p>
```

Cabe destacar que el ámbito de aplicación de este atributo afecta a todos sus hijos. Es decir, que al estar el elemento SPAN (de la ilustración anterior) dentro de un elemento de párrafo editable, también se volverá editable.

Como nota adicional añadiremos que, desde JavaScript, es posible conocer si el elemento tiene esta propiedad establecida a través del método `ISCONTENTEDITABLE`.

2.4.6 Atributo draggable

El atributo DRAGGABLE especifica que el elemento indicado puede ser arrastrado a un contenedor.

```
<p draggable="true">Este es un párrafo arrastable</p>
```

NOTA

Cuando se define un atributo DRAGGABLE, también es importante definir el atributo TITLE para proporcionar un identificador único si se realizan interacciones a través de herramientas de asistencia. Además, en el elemento origen se debe declarar el evento ONDRAGSTART y, en el objeto destino, los eventos ONDROP y ONDRAGOVER.

2.4.7 Atributo dir

El atributo DIR especifica la direccionalidad del texto, es decir, si los contenidos están escritos en un idioma que se lee de izquierda a derecha, o de derecha a izquierda.

```
<p dir="rtl">
Texto legible de derecha a izquierda para idiomas como el árabe.
</p>
```

2.4.8 Atributo enterkeyhint

El atributo ENTERKEYHINT especifica el texto o icono de acción que se debe representar como efecto de aceptación (salto de línea o a siguiente campo) en los teclados virtuales. Por tanto, sólo podrá ser definido en controles de formulario como el elemento TEXTAREA e INPUT, o en aquellos elementos que tengan establecida la propiedad CONTENTEDITABLE a true.

```
<input enterkeyhint="search">
```


Entre los posibles valores que puede tomar este atributo, tenemos las opciones de ENTER, que muestra el icono o texto de “ingresar”, DONE, que muestra el icono o texto de

“hecho”, **GO**, que muestra el icono o texto de “ir”, **NEXT**, que muestra el icono o texto de “siguiente”, **PREVIOUS**, que muestra el icono o texto de “anterior”, **SEARCH**, que muestra el icono o texto de “buscar” y, **SEND**, que muestra el icono o texto de “enviar”.

2.4.9 Atributo hidden

El atributo **HIDDEN** especifica que el elemento no es relevante y, como consecuencia, no debe mostrarse en pantalla. Esto puede ser útil cuando se desea que los elementos del documento, o de una sección de este, se vuelvan visibles en función de una condición o circunstancia determinada.

```
<p hidden>Este párrafo permanecerá oculto hasta nueva orden.</p>
```

 **NOTA**

Cabe destacar que, la visualización u ocultación de elementos también es posible realizar a través de CSS, mediante la propiedad **DISPLAY**.

2.4.10 Atributo inputmode

El atributo **INPUTMODE** sugiere, o proporciona pistas, sobre el tipo de datos que se pueden introducir en los elementos editables. Por tanto, sólo podrá ser definido en controles de formulario, como los elementos **INPUT** y **TEXTAREA**, o en aquellos elementos que tengan establecida la propiedad **CONTENTEDITABLE** a **true**.

Sus posibles valores son:

Valor	Descripción y ejemplo
none	Indica que no tiene teclado virtual y que la página implementará su propio control de entrada.
text	Indica que se utilizará la entrada estándar definida.
decimal	Indica que la entrada será un valor numérico con decimales separados por coma o punto. La visualización del signo negativo dependerá del dispositivo donde se reproduzca.
numeric	Indica que la entrada será un valor numérico entero. La visualización del signo negativo dependerá del dispositivo donde se reproduzca.
tel	Indica que la entrada permitirá los dígitos del 0 al 9, el * y #.
search	Indica que el teclado virtual debe optimizarse para la búsqueda.
email	Indica que la entrada es de tipo email.
url	Indica que la entrada es de tipo URL.

```
<div contenteditable="" inputmode="search"></div>
```

2.4.11 Atributo id

El atributo ID especifica un identificador que vincula la etiqueta HTML con ese nombre. Dada la importancia que tienen para los lenguajes de script, la accesibilidad web, la usabilidad web y el posicionamiento SEO, los identificadores de etiqueta no deben repetirse bajo el mismo contexto.

```
<label>  
Nombre completo  
<input id="name" />  
</label>
```

2.4.12 Atributo lang

El atributo LANG especifica el idioma en el que está descrito el contenido del elemento. La anotación para indicar el idioma debe ser representado bajo el estándar ISO-639-1, el cual identifica el lenguaje a partir de dos caracteres.

```
<p lang="es">Mi nombre es Pablo</p>  
<p lang="en">My name is Paul</p>
```

Dado que HTML tiene un carácter hereditario, su aplicación afectará al propio elemento y sus hijos, al igual que sucede cuando lo declaramos en la etiqueta HTML, que indica que todo el documento, por tanto, todos sus elementos, están definidos en ese idioma.

2.4.13 Atributo nonce

El atributo NONCE especifica un código criptográfico de uso único que puede ser utilizado por la Política de Seguridad de Contenido para indicar si el elemento va a ser cargado y aplicado al documento actual o si se permitirá que se continúe con una acción determinada.

Dicho de otra forma, el atributo NONCE resulta ser una manera de decirle a los agentes de usuario que el elemento SCRIPT o LINK que está definido en el documento, no fue inyectado por un tercero (malicioso), sino que fue colocado de forma intencionada en el documento por un usuario legítimo que puede manipular el servidor.

```
<link rel="search" href="booksearch.asp"  
nonce="bGlicm8tY29uc3J0dWNjac0zbi1kZS1wYWdpbmFzLXd1Yg==">
```

El código de este atributo, en general, se genera en el servidor, aunque resulta posible obtener un valor legítimo a través de las herramientas online que están disponibles en Internet. Su valor se corresponde con una codificación en Base64 de, al menos, 128 bits.

Para indicarle al servidor que ese NONCE es legítimo, en el encabezado de las peticiones se debe declarar una CSP (Política de Seguridad de Contenido) como la siguiente:

```
Content-Security-Policy: script-src 'nonce-  
bGlicm8tY29uc3J0dWNjac0zbi1kZS1wYWdpbmFzLXd1Yg=='
```

Esto ayudará a evitar ciertos ataques, incluyendo los famosos ataques XSS (Cross-Site Scripting) y ataques por inyección de datos.

2.4.14 Atributo slot

El atributo SLOT especifica o asigna un espacio dentro del árbol Shadow DOM de un componente web. Como dato importante, el valor de este atributo debe coincidir exactamente con el atributo NAME del elemento que define el atributo SLOT.

```
<span slot="book">Domine JavaScript 4ª Edición</span>
```

La creación y definición de Web Components está descrita en varios sitios web y, también, explicada en detalle en el libro “Domine JavaScript 4ª Edición” de esta misma editorial.

No obstante, esta especificación todavía está en fase experimental y, por tanto, es posible que se requiera de algún polyfill escrito en un lenguaje como JavaScript para poder utilizarlo.

2.4.15 Atributo style

El atributo STYLE especifica las propiedades de estilo para un elemento o conjunto de ellos. Aunque es altamente aconsejable que las propiedades de estilo se definan en documentos CSS independientes, esta etiqueta puede ser útil para declarar estilos de uso específico o para realizar pruebas.

```
<h1 style="margin-top: 5px;">Título con margen superior</h1>
```

2.4.16 Atributo tabindex

El atributo TABINDEX especifica si el elemento puede obtener el foco y su posición relativa en la secuenciación de obtención del foco.

Por ejemplo, si su valor es -1, el elemento no podrá tomar el foco. Los elementos no modificables suelen tener esta opción definida por defecto.

Si su valor es 0, podrá tomar el foco, pero por orden de llegada, es decir, por el orden que se establezca según se van declarando los elementos. Los elementos modificables tienen esta opción definida por defecto.

Ahora bien, si su valor es mayor que 0, lo que se indica al agente de usuario es que tiene prioridad con respecto a los elementos que tengan un TABINDEX menor.


```
<p tabindex="0">
Este párrafo puede tomar el foco
</p>
```

2.4.17 Atributo title

El atributo TITLE especifica el mensaje, a modo de tooltip, que deberá ser mostrado cuando se sitúe el dispositivo puntero sobre dicho elemento.

```
<p title="Esto es un párrafo de prueba!">
Esto es un párrafo con un mensaje emergente
</p>
```

2.4.18 Atributo translate

El atributo TRANSLATE especifica si el valor contenido dentro de ese elemento debe ser traducido o no. Sus posibles valores son **YES** y **NO**, no obstante, el único que de verdad se usa es **NO** ya que es el que indica a las herramientas de traducción que el texto no debe ser traducido.

```
<p>Visita www.<span translate="no">Islavisual</span>.com</p>
```

2.4.19 Atributos personalizados

Los atributos personalizados podrían definirse como un contenedor de datos modificables que permiten su manipulación e integración con el DOM de los lenguajes de script como JavaScript.

Una de las razones por las que se dotó a HTML5 de esta capacidad es porque, hasta entonces, el almacenamiento de información asociada a elementos independientes se hacía a través del atributo CLASS, lo que muchas veces no resultaba muy elegante ni claro.

Hoy día, los atributos personalizados son útiles en muchas ocasiones. Por ejemplo, imaginemos una situación en la que se está ofreciendo una lista de elementos seleccionables a través de una tabla. En esta supuesta tabla, se están mostrando datos al usuario, pero no se desea mostrar el identificador de registro. Sin embargo, para conocer qué registro o elemento seleccionó el usuario, debemos recurrir a ese ID.

Pues bien, se puede almacenar en un atributo personalizable y, más tarde, con alguna funcionalidad creada en un lenguaje de script como JavaScript, recuperar ese ID y enviárselo al servidor a través de Ajax.

La forma de definir un atributo personalizado es mediante el prefijo DATA- y un sufijo que no puede empezar con la palabra XML, ni contener puntos, comas o mayúsculas.

```
<p id="p1" data-id="002">
Esto es un párrafo con un atributo personalizado
</p>
```

A modo informativo, también se puede recuperar el atributo personalizado a través de un lenguaje de script como JavaScript. La forma de recuperarlo es mediante el uso de la propiedad DATASET, la cual proporciona una interfaz para manipular los atributos personalizados definidos en HTML5. En este caso, para acceder al atributo anteriormente expuesto, podríamos hacer lo siguiente:

```
<script type="text/javascript">
document.querySelector("#p1").dataset.id;
</script>
```

2.5 PRACTICA Y JUEGA

Juego: Tags ID 1 / Elementos básicos de HTML



El juego repasa las etiquetas HTML5 y tiene como objetivo seleccionar la etiqueta correcta en cada caso con el mínimo número de errores y en el menor tiempo posible. Se puede acceder desde la dirección <https://codepen.io/pefc/full/NWLgMmQ>

Ejercicio:

Definir un documento HTML5 con todas las declaraciones y metadatos necesarios para crear una página web estándar. Como parte importante deberemos asegurarnos de que:

- La codificación de caracteres en Unicode de 8-bit.
- Estén presentes el título del documento.
- La definición del área útil o ventana de visualización.
- La hoja de estilos de nuestro documento, la cual llamaremos “styles.css”.
- El documento de JavaScript con las funciones personalizadas de nuestro documento, al cual llamaremos “scripts.js”.

Estructura de un documento HTML

Definir la estructura básica para un documento cualquiera de HTML. En la siguiente dirección se dispone de una posible respuesta.

<https://codepen.io/pefc/pen/oNPBope>

Código QR

