1

LOS LENGUAJES DE MARCADO

1.1. LENGUAJES DE MARCADO

Los lenguajes de marcado, también se les suele llamar lenguajes de marcas, aunque este tipo de referenciación no es nada habitual.

Históricamente, los lenguajes de marcado eran utilizados por las editoriales y medios de comunicación para imprimir instrucciones concretas en los márgenes. Para ello, utilizaban unos "marcadores" que indicaban el tipo de fuente, estilo, tamaño de letra y la corrección de errores. Una cosa llevó a la otra y, con el tiempo, se estandarizaron como un grupo de marcas que, más tarde, fueron reinterpretadas en lo que hoy conocemos como lenguajes de marcado.

El lenguaje de marcado es un término que hace referencia a una manera de codificar la información. Esta codificación se caracteriza porque los contenidos están envueltos y declarados a través de unas entidades que llamaremos etiquetas o marcas.

En este contexto, cuando se habla de etiquetas o marcas, además se les suele dar un significado que puede indicar qué es o qué debe contener, dónde utilizarlo o cómo presentarlo. No obstante, ninguna de estas marcas tiene la capacidad de ejecutar acciones ni de realizar operaciones de ninguna clase.

Esto último debe tenerse claro porque, en ocasiones, los lenguajes de marcado se confunden con los lenguajes de programación, y nada más lejos. Un lenguaje de programación es un idioma de computación que presenta varias peculiaridades como

son la posibilidad de declarar variables, realizar operaciones matemáticas o crear fragmentos de código que suelen ser agrupados en procedimientos o funciones. Un lenguaje de marcado no hace nada de eso, sólo etiqueta la información aportándole un valor semántico.

1.1.1. Características de los lenguajes de marcado

Un lenguaje de marcado se caracteriza porque tiene marcas o etiquetas que son declaradas a modo de elemento o entidad. Cada uno de estos elementos o entidades tiene un contenido y uno o varios atributos que definen su comportamiento, su función y su significado.

Un ejemplo de todo esto podría ser la declaración de un producto.

```
<producto>
    <cabecera>
        <titulo>Nombre producto</titulo>
        <subtitulo>Categoría</subtitulo>
    </cabecera>

    <descripcion>
        <texto>Descripción del producto</texto>
    </descripcion>

    <pie>
        <negrita>Precio</negrita>
        <accion>Comprar</accion>
    </pie>
</producto>
```

Ilustración 1.11.1. Ejemplo de lenguaje de marcado

Como se puede observar en la ilustración anterior, se ha definido una entidad concreta a través de unos elementos que definen su significado y comportamiento y que, en ocasiones, tienen propiedades intrínsecas que aportan un valor a la presentación.

Un ejemplo de ello podría ser la etiqueta SUBTITULO, que indica que, su contenido textual, es un texto de cabecera o a modo de título y que está en el segundo nivel de la jerarquía en lo referente a títulos.

Si, además, este código fuese interpretado por una herramienta o sistema específico, podría llevar una interpretación específica que provocase que fuese presentado de forma concreta, con un tamaño, margen y estilo concretos, entre otras cosas.

1.1.2. Clasificación de los lenguajes de marcado

Los lenguajes de marcado suelen ser clasificados en tres grupos, no obstante, pueden llegar a mezclarse propiedades de varios en uno, formando nuevas especificaciones.

1.1.2.1. Marcado de presentación

El marcado de presentación es aquel que está enfocado hacia la presentación de la información del documento, es decir, es aquel que sirve para dar formato al texto. Este tipo de marcado es suficiente para el proceso de lectura, pero insuficiente para el procesamiento automático de la información. Además, las etiquetas de marcado están ocultas al usuario, lo que hace que su contenido esté ofuscado y sea complicado de extraer si no es con la herramienta adecuada.

El marcado de presentación puede ser sencillo de elaborar, sin embargo, su mantenimiento y/o modificación pueden volverse complicados, por lo que su uso no está demasiado extendido.

Un ejemplo de marcado de presentación es RTF, en donde la composición de las etiquetas suele proporcionar un comportamiento y/o un significado distinto. Es decir, el título del documento podría venir predefinido por una combinación de varios saltos de línea en cualquier posición del documento y terminar con un carácter especial justo detrás del contenido textual.

```
{\rtf1\ansi\ansicpg1252\deff0\deflang3082
{\fonttbl{\f0\fswiss\fcharset0 Arial;}{\f1\fmodern\fprq1\fcharset0 Arial;}}
{\colortbl ;\red255\green0\blue0;\red0\green0\blue128;}
\viewkind4\uc1\pard\b\f0\fs20 RTF\b0 es un \cf1\b\fs24 lenguaje de marcado
de presentaci\'f3 \f0\fs20 .\par}
```

Ilustración 1.11.2.1. Fragmento de código de documento RTF

1.1.2.2. Marcado de procedimientos

El marcado de procedimientos es aquel que está enfocado hacia la presentación del texto, al igual que sucede con el marcado de presentación. No obstante, las marcas o etiquetas que componen el documento no están ocultas, es decir, son visibles en todo momento para el usuario.

El marcado de procedimientos es sencillo de elaborar y de modificar. Además, es interpretado por orden de aparición, por lo que ayuda a predecir cómo será su resultado durante el proceso de lectura.

Un ejemplo de marcado de procedimientos es LaTeX y HTML. Ambos requieren de una marca o etiqueta que funciona a modo de identificador especial que le indica al sistema cómo se debe renderizar el contenido textual. Este renderizado suele venir definido a través de una serie de instrucciones que establecen el texto con un tipo de fuente, tamaño, alineación y estilos concretos.

```
<article>
    <header>
        <h2>Nombre producto</h2>
    </header>

    <div class="descripcion">
        <p>Descripción del producto</p>
    </div>

    <footer>
        <b>Precio</b>
        <button>Comprar</button>
    </footer>
</article>
```

Ilustración 1.11.3. Fragmento de código de documento HTML5

1.1.2.3. Marcado descriptivo

El marcado descriptivo, también conocido como semántico, es aquel que utiliza las etiquetas para describir fragmentos de texto, pero sin especificar en qué orden o cómo deben representarse.

El marcado descriptivo suele ser sencillo de elaborar y modificar y, a menudo, sus etiquetas pueden ir acompañadas de atributos que definen su significado o comportamiento, pero su interpretación está separada de la presentación, es decir, sólo especifica la descripción del tipo y contenido de los documentos.

Al marcado descriptivo también se le atribuye la cualidad de simplificar el proceso de reformateado del texto, principalmente, porque la información del formato está separada del propio contenido. Por esta razón, un fragmento indicado como cursiva a través de la etiqueta I (italic), podría emplearse para marcar énfasis o para indicar un contenido gráfico.

Cabe destacar que, si quisiéramos sortear esta última situación en el marcado de presentación y en el marcado de procedimientos, el proceso probablemente sería bastante tedioso, sin embargo, si los contenidos se hubiesen diferenciado descriptivamente mediante etiquetas distintas, podrían representarse de manera diferente sin esfuerzo.

Un ejemplo de marcado descriptivo es SGML y el XML. Ambos son muy flexibles ya que los fragmentos se etiquetan como son y no como deberían mostrarse.

```
<!doctype email system "email.dtd">
<email>
    <to>info@ejemplo.com
    <from>alumno@gmail.com
    <date>mon, 29 jan 2020 12:00:02 - 0100 (est)
    <subject>error de acceso
    <contents>
        Hola, buenas tardes. ¿Mi cuenta está bloqueda o deshabilitada?.
        <url>http://ejemplo.com</url>
        saludos
    </contents>
</email>
```

Ilustración 1.4. Fragmento de código de documento SGML

1.2. INTRUDUCCIÓN A HTML5

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.

1.3. ELEMENTOS BÁSICOS DE HTML

1.3.1. Definición del tipo de documento DTD (!DOCTYPE)

Cuando uno decide trabajar con HTML, lo primero que debe hacer es declarar es 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.

1.3.1.1. DTDs de HTML

HTML5

```
<!DOCTYPE html>
```

1.3.1.2. DTDs de MathML

MathML 2.0

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

1.3.1.3. 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">
```

1.3.1.4. DTDs de XHTML

XHTML1.1

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

1.3.2. 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>
```

1.3.3. 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.

1.3.4. Etiqueta body

La etiqueta BODY es el elemento o la estructura que se 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>
```

1.3.5. 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 -->
```

1.4. INFORMACIÓN DEL DOCUMENTO

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

1.4.1. Elemento base

El elemento BASE especifica la base de direccionamiento predeterminada para los elementos que utilicen un direccionamiento relativo, es decir, las direcciones que 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>
```

1.4.2. 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	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. 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. `<link rel="dns-prefetch" href="//fonts.googleapis.com"/>`
prefetch / preload	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. 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. 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. `<link rel="preload" href="./js/scripts.js" as="script">`
preconnect	La cláusula PRECONNECT indica al agente de usuario que resuelva el DNS y realice la preconexión con los protocolos TCP y TLS, si procede. 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. `<link rel="preconnect" href="//islavisual.com/">`

prerender	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. 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 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. `<link rel="preconnect" href="//islavisual.com/">`

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.

1.4.3. 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.

1.4.3.1. Atributos del elemento meta

1.4.3.1.1 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">
```

1.4.3.1.2 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 procesado, accesibilidad y/o usabilidad web.

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

1.4.3.1.3 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">
```

1.4.3.1.4 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">
```

1.4.3.1.5 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">
```

1.4.4. 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.

1.4.5. 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.

1.4.6. 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>
```

1.4.7. 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 re-fiere al área útil de la pantalla donde 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 pixeles 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.