Fundamentos de HTML

PID_00253485

Mark Norman Francis
Christian Heilmann
Roger Johansson

Tiempo mínimo previsto de lectura y comprensión: 2 horas

Universitat Oberta de Catalunya

Índice

1. Conceptos básicos de HTML

Mark Norman Francis

En este apartado aprenderemos los conceptos básicos de HTML: qué es, qué hace, un resumen de su historia y qué estructura tiene un documento en HTML. Los apartados que encontraréis después de éste explican cada una de las partes específicas de HTML con mucho más detalle.

1.1. Qué es el HTML

La mayoría de las aplicaciones de escritorio que pueden leer y escribir ficheros utilizan un formato de fichero especial. Por ejemplo, Microsoft Word entiende los ficheros .doc y Microsoft Excel entiende los .xls. Estos ficheros contienen las instrucciones para reconstruir el documento cuando se vuelve a abrir y para saber cuál es su contenido, además de los "metadatos" sobre el artículo, como por ejemplo el autor, la fecha de la última modificación del documento e incluso cosas como la lista de cambios realizados con el fin de poder recuperar todas sus diferentes versiones.

El **HTML** (**HyperText Markup Language**) es un lenguaje para describir el contenido de los documentos de la web. Utiliza una sintaxis especial que contiene marcadores (conocidos como "elementos") que rodean al texto que hay dentro del documento para indicar a los agentes de usuario cómo deben interpretar esta parte del documento.

Utilizamos el término técnico *agentes de usuario* en lugar de *navegadores web*. Un agente de usuario es cualquier software que se utilice para acceder a las páginas web en nombre de los usuarios. Aquí debemos hacer una distinción muy importante: todos los tipos de navegadores de escritorio (Internet Explorer, Opera, Firefox, Safari, etc.) y los navegadores alternativos para otros dispositivos (como el canal de Internet de la Wii y los navegadores para teléfonos móviles como Opera Mini y WebKit para el iPhone) son agentes de usuario, pero no todos los agentes de usuario son navegadores. Los programas automatizados que utilizan Google y Yahoo! para indexar la web y que ejecutan en sus motores de búsqueda también son agentes de usuario, pero no hay ningún ser humano que los controle directamente.

1.2. Aspecto del HTML

El HTML es sólo una representación textual del contenido y de su significado general. Por ejemplo, el código para el encabezamiento "Aspecto del HTML" es el siguiente:

```
<h2 id="htmllooks">Aspecto del HTML</h2>
```

La parte `<h2>` es un marcador (que se conoce como "etiqueta") que significa "lo que sigue se debe considerar como un título de segundo nivel". `</h2>` es una etiqueta que indica dónde acaba el título de segundo nivel (y se conoce como "etiqueta de cierre"). La etiqueta de apertura, la etiqueta de cierre y todo lo que hay entre ellas se conoce como "elemento". Mucha gente utiliza los términos *elemento* y *etiqueta* indistintamente, lo cual no es del todo correcto. `id="htmllooks"` es un atributo; ya hablaremos de los atributos más adelante.

La mayoría de los navegadores incorporan una opción "Código fuente" o "Ver el código fuente", normalmente bajo el menú "Ver". Si vuestro navegador incorpora esta opción, seleccionadla y dedicad unos momentos a mirar el código fuente HTML de esta página.

1.3. La historia del HTML

Cuando Tim Berners-Lee inventó la web, creó tanto el primer servidor web como la primera versión del HTML*.

*** http://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/MarkUp.html**

El HTML ha cambiado de una manera muy importante desde aquellos primeros días, pero una buena parte del contenido del HTML actual ya se encontraba en esta primera documentación y más de la mitad de las "etiquetas" descritas en el documento "Etiquetas HTML" original todavía existen.

En el apartado 1 del módulo "Introducción al mundo de los estándares web" visteis resumidamente la aparición de la web moderna.

A medida que fue habiendo cada vez más gente que escribía páginas web y alternativas al navegador original, también se fueron añadiendo más funciones a HTML. Muchas se adoptaron universalmente (como el elemento img utilizado para insertar una imagen en un documento, que se utilizó por primera vez en NCSA Mosaic). Otras etiquetas eran propias y sólo se utilizaban en uno o dos navegadores. Cada vez había una necesidad mayor de estandarización a fin de que los autores de otros navegadores pudieran tener un documento (conocido como "especificación") que describiera de manera definitiva el aspecto de HTML para así poder juzgar si se saltaban algunas partes de la implantación de HTML.

El IETF (Internet Engineering Task Force, un organismo definidor de estándares que se preocupa por la interoperabilidad a través de Internet) publicó en el año 1993 un borrador de propuesta de HTML. Esta propuesta caducó en el año 1994 sin haberse convertido en un estándar, pero provocó que el IETF creara un grupo de trabajo para estudiar la estandarización del HTML.

En el año 1995 se redactó "HTML 2.0", que aprovechaba ideas del borrador de HTML original. Dave Raggett escribió una propuesta alternativa conocida como HTML+; esta propuesta se utilizó como base para muchos de los elementos nuevos implantados por los navegadores (como el método para insertar imágenes en documentos, que se utilizó por primera vez en NCSA Mosaic).

Más tarde, en aquel mismo año apareció un borrador de HTML 3.0, pero se dejó de trabajar en esta versión a causa de la falta de apoyo por parte de los creadores de navegadores en cuanto a la dirección que se debía seguir. HTML 3.2 suprimió muchas de las funciones nuevas de la versión 3.0 y, en lugar de éstas, adoptó muchas de las creaciones de los navegadores más populares de entonces: Mosaic y Netscape Navigator.

En el año 1997, el W3C publicó HTML 4.0 como recomendación que adoptaba más extensiones específicas de navegadores, pero que también intentaba racionalizar y perfeccionar HTML. Esto se hizo marcando varios elementos como desaprobados (*deprecated*), que significa que los elementos están obsoletos y que, aunque todavía aparecen en esta versión, se eliminarán en una revisión posterior. Esto se hizo para impulsar un uso mejor y más semántico de HTML en documentos.

Podéis ver la evolución de HTML en el apartado 3 del módulo "Introducción al mundo de los estándares web".

En el año 1999 se publicó HTML 4.01, con algunas erratas, que se corrigieron en el año 2001. En el año 2000, el W3C también publicó la especificación XHTML 1.0, que era HTML reestructurado para ser un documento XML válido.

En el año 2004 el *Web Hypertext Application Technology Working Group* (WHATWG*) inició el trabajo para desarrollar la especificación de HTML5. En enero de 2008 se publicó el primer "borrador de trabajo" de HTML5**, que posteriormente alcanzaría el estatus de "candidate recommendation" en diciembre de 2012. El documento lo mantiene el HTML Working Group*** del W3C.

*** https://whatwg.org/**
**** https://www.w3.org/TR/html5/**
***** https://www.w3.org/html/wg/**

1.4. La estructura de un documento HTML

El documento HTML5 válido más pequeño posible sería algo similar a esto:

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="utf-8">
    <title>Página de ejemplo</title>
  </head>
  <body>
    <h1>Hola mundo</h1>
  </body>
</html>
```

El documento empieza con un elemento de tipo de documento o `doctype`. Este elemento describe el tipo de HTML que se utiliza para que los agentes de usuario puedan determinar cómo se debe interpretar el documento y saber si sigue las normas que dice que seguirá.

El `doctype` se describe con más detalle en el apartado 3 de este módulo.

A continuación, se puede ver la etiqueta de apertura del elemento `html`. Éste es un elemento que incluye a todo el documento. La etiqueta `html` de cierre es lo último que hay en cualquier documento HTML.

Dentro del elemento `html` está el elemento `head`. Éste es un elemento que contiene información sobre el documento (los metadatos). Este elemento se describe más detalladamente en el apartado siguiente. En la cabecera está el elemento `title`, que define el título "Página de ejemplo" de la barra del menú.

Después del elemento `head` está el elemento `body`, que es el elemento que incluye el contenido real de la página; en este caso sólo hay un elemento de encabezamiento de nivel uno (`h1`), que contiene el texto "Hola a todos". Y éste es todo nuestro documento.

Como se puede ver, a menudo los elementos contienen otros elementos. El cuerpo del documento incluirá inevitablemente muchos otros elementos. Las divisiones de página crean la estructura general del documento y contendrán subdivisiones. Éstas contendrán títulos, párrafos, listas, etc. Los párrafos pueden contener elementos que creen enlaces hacia otros documentos, citas, énfasis, etc. A medida que vayáis avanzando en esta serie, iréis sabiendo más cosas sobre estos elementos.

1.5. La sintaxis de los elementos HTML

Como ya hemos visto, un elemento básico en HTML consiste en dos marcadores al principio y al final de un bloque de texto. Hay algunos elementos que no rodean al texto y, en la mayoría de los casos, los elementos pueden contener subelementos (como `html`, que contiene `head` y `body` en el ejemplo anterior).

Los elementos también pueden tener atributos, que pueden modificar el comportamiento del elemento e introducir un significado adicional.

```
<div id="masthead">
   <h1>Conceptos básicos del
      <abbr title="lenguaje de marcado de hipertexto">HTML</abbr>
   </h1>
</div>
```

En este ejemplo hay un elemento `div` (división de página, una manera de partir los documentos en bloques lógicos) con un atributo `id` añadido que tiene el valor de `masthead`. El elemento `div` contiene un elemento `h1` (encabezamiento de primer nivel o más importante), que al mismo tiempo contiene texto. Parte de este texto está incluido en un elemento `abbr` (que se utiliza para especificar la expansión de las siglas), que tiene el atributo

`title`, cuyo valor está definido como lenguaje de marcado de hipertexto (`Hypertext Markup Language`).

Muchos de los atributos de HTML son comunes para todos los elementos, aunque algunos son específicos de uno o varios elementos concretos. Éstos tienen siempre la forma `keyword="value"`. El valor debe aparecer siempre entre comillas simples o dobles (en algunas circunstancias se pueden omitir las comillas, pero no es una práctica recomendable con respecto a la predictibilidad, la comprensión y la claridad; se deben poner siempre los valores entre comillas).

La mayoría de los atributos y sus valores posibles están definidos por las especificaciones HTML*; no es posible crear atributos propios sin invalidar el HTML, ya que ello puede confundir a los agentes de usuario y provocar problemas a la hora de interpretar correctamente la página web. Las únicas excepciones reales son los atributos `id` y `class`; sus valores están totalmente bajo vuestro control, ya que sirven para añadir significado y semántica propias a vuestros documentos.

*** http://www.w3.org/TR/html401/index/attributes.html**

Un elemento que se encuentra dentro de otro elemento se conoce como "hijo" de este elemento. Así pues, en el ejemplo anterior, `abbr` es hijo del elemento `h1`, que al mismo tiempo es hijo de `div`. Y al revés, el elemento `div` sería "padre" del elemento `h1`. Este concepto de padre/hijo es muy importante, ya que es la base de CSS y se utiliza mucho en JavaScript.

1.6. Elementos de bloque y en línea

En el HTML hay dos categorías generales de elementos que corresponden a dos tipos de contenidos y estructuras que representan estos elementos: elementos de bloque y elementos en línea.

Los **elementos de bloque** son elementos de nivel superior y normalmente definen la estructura del documento. Puede ser útil ver los elementos de bloque como aquellos que empiezan en una línea nueva y que representan una ruptura con lo anterior.

Algunos elementos de bloque comunes incluyen los párrafos, las listas, los títulos y las tablas.

Los **elementos en línea** son aquellos que se encuentran incluidos en los elementos estructurales de bloque y que incluyen sólo partes pequeñas del contenido del documento, y no párrafos enteros ni grupos de contenido.

Un elemento en línea no hará que aparezca una línea nueva en el documento; son los tipos de elementos que aparecen dentro de un párrafo de texto. Algunos elementos en línea comunes son los vínculos de hipertexto, las palabras o frases destacadas o las citas breves.

1.7. Referencias de carácter

Un último aspecto que hay que mencionar de un documento HTML es la manera de incluir caracteres especiales. En el HTML, los caracteres <, > y & son especiales. Estos caracteres inician y finalizan partes del documento HTML, y no representan los caracteres de "menor que", "mayor que" y "et".

Uno de los primeros errores que puede cometer un autor de webs es utilizar un carácter "et" en un documento y entonces ver que aparece algo inesperado. Por ejemplo, si se escribe la frase "Imperial units measure weight in stones&pounds" puede ser que en algunos navegadores se vea "...stones£s".

Esto sucede porque en HTML la cadena literal "&pound" es en realidad una referencia de carácter. Una referencia de carácter es una manera de incluir en un documento un carácter que es difícil o imposible de escribir desde un teclado, o en una codificación de documento concreta.

El símbolo "et" (&) introduce la referencia, y el punto y coma (;) la acaba. No obstante, muchos agentes de usuario pueden ser muy condescendientes a la hora de perdonar errores en el HTML, como por ejemplo el de olvidar el punto y coma, e interpretarán "&pound" como una referencia de carácter. Las referencias pueden ser números (referencias numéricas) o palabras abreviadas (referencias de entidades).

Un símbolo "et" se debe introducir en un documento como "&", que es la referencia de entidad de carácter, o como "&", que es la referencia numérica. En la web del W3C encontraréis una tabla completa de las referencias de caracteres*.

*** https://dev.w3.org/html5/html-author/charref**

Resumen

En este apartado hemos aprendido los conceptos básicos del HTML, de dónde proviene y algunas ideas sobre la estructura de un documento HTML. A continuación, explicaremos la sección `<head>` de un documento HTML más detalladamente y después continuaremos explicando el contenido de `<body>`.

2. El elemento <head> de HTML

Christian Heilmann

Este apartado trata una parte del documento HTML que no recibe la atención que se merece: el marcado que va dentro del elemento `head`. Cuando acabéis este tutorial, habréis aprendido las diferentes partes de esta sección y qué hacen, incluyendo el `doctype`, el elemento `title`, las palabras clave y la descripción (todos ellos gestionados con elementos `meta`). También aprenderéis los detalles de JavaScript y de los estilos CSS (tanto internos como externos) y sabréis qué no debéis dejar dentro de `head`. Podéis descargaros algunos ficheros de demostración*, a los que haremos referencia a lo largo de este apartado; una vez hayáis acabado de trabajar, podéis dedicaros a modificarlos como queráis. Debéis seguir este tutorial de principio a fin, ya que va presentando una serie de buenas prácticas que se deben seguir cuando se trabaja con el `head` del HTML. Aunque todas las partes son válidas en sí mismas, al final encontraréis una conclusión sobre las mejores prácticas que os hará reconsiderar algunos de los primeros consejos.

*** http://mosaic.uoc.edu/ac/le/operafiles/headtutorial.zip**

2.1. ¿Head? ¿Qué es eso?

En esta misma asignatura ya habéis visto que un documento HTML válido necesita un `doctype`, que es un elemento que explica qué tipos de HTML hay que esperar e indica a los navegadores la manera de mostrar correctamente el documento. Roger explica mucho más detalladamente el elemento `doctype` en el apartado siguiente, aquí nos limitaremos a decir que el `doctype` especifica que el documento necesita un elemento `html` con unos elementos `head` y `body` en su interior. La mayor parte del tiempo la dedicaréis al elemento `body`, ya que es el que incorpora todo el contenido del documento. `head` tiene un papel aparentemente menos importante, ya que, si dejamos de lado el elemento `title`, no hay nada de todo lo que ponéis dentro de esta sección que sea visible para las personas que visitan vuestra web. Pero `head` es el lugar donde están la mayor parte de las instrucciones para el navegador y donde se guarda información adicional (la que se conoce como información `meta`) sobre el documento.

2.2. Definir el idioma principal del documento

Una parte de la información sobre el documento se encuentra en el elemento padre de `head`, que es el elemento `html`. Aquí se define el idioma natural principal del documento. Cuando hablamos de idioma natural nos

estamos refiriendo a un idioma *humano*, como por ejemplo el francés, el tailandés o el inglés. Esto es de gran ayuda para los lectores de pantalla, ya que, por ejemplo, la palabra *six* se pronuncia de una manera muy diferente en francés y en inglés; también puede ser de mucha ayuda para los motores de búsqueda. Es muy aconsejable definir el idioma principal de un documento, sobre todo si estáis escribiendo páginas pensadas para un público internacional, aunque hay muchas páginas que no lo hacen. Se debe definir de la manera siguiente:

```
<html lang="en-GB">
   ...
</html>
```

Tened en cuenta que también podéis definir el idioma de subsecciones de vuestro documento utilizando el atributo `lang` en otros elementos, como por ejemplo `<span lang="fr">Bonjour</span>`.

Los códigos de idioma pueden ser códigos de dos letras, como por ejemplo `en` para el inglés, o códigos de cuatro letras como `en-US` para el inglés americano u otros códigos menos habituales. Los códigos de dos letras están definidos en ISO 639-1*, aunque las mejores prácticas actuales exigen el uso del registro de subetiquetas de la IANA para las definiciones de los códigos de idioma**.

*** http://www.w3.org/International/articles/language-tags/**

**** http://en.wikipedia.org/wiki/ISO_639-1**

2.3. Juzgar un documento por su título

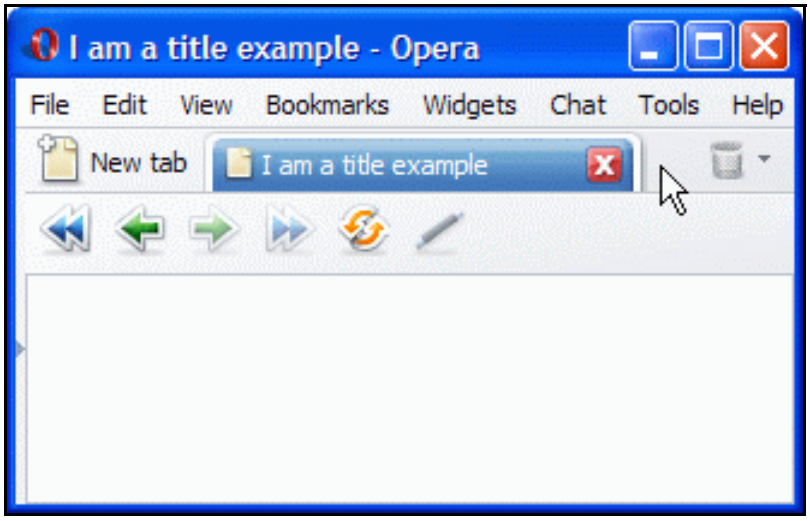
Uno de los elementos más importantes de `head` es `title`. El texto incluido en el `title` aparece en la mayoría de los agentes de usuario/navegadores dentro de la barra del título de la aplicación (la barra que hay en la parte superior de la ventana del navegador). Es lo primero que verán los usuarios de la web cuando visiten vuestro sitio; por lo tanto, es realmente importante. Además, las tecnologías de asistencia como por ejemplo los lectores de pantalla (software que lee en voz alta las páginas web para los usuarios con alguna discapacidad visual) lo leen para dar una primera idea sobre lo que los visitantes pueden esperar del documento, y también hay muchos motores de búsqueda que funcionan de un modo similar. Así pues, si utilizáis un buen título que sea legible para las personas y que contenga las palabras clave adecuadas, tendréis muchísimas más posibilidades de que os encuentren en la Red. Ahora seleccionaremos el documento HTML `headexample.html*` y lo abriremos en el navegador.

*** http://mosaic.uoc.edu/ac/le/operafiles/headexample.html**

```
<!DOCTYPE html>
<html>
<head>
  <title>I am a title example</title>
</head>
<body>
</body>
</html>
```

Veréis que el texto de `title` aparece en la barra que hay sobre el navegador, tal como muestra la figura 1.

Figura 1



Mostrar un título en un navegador

Hay muchos tutoriales en la web que explican cómo escribir unos buenos títulos de documentos; en la mayoría de los casos, las explicaciones están relacionadas con la optimización para motores de búsqueda (SEO, del inglés *search engine optimization*). Tampoco hay que exagerar e intentar engañar a los motores de búsqueda para que muestren un número excesivamente alto de resultados de búsqueda. Debéis escribir un título que dé una información concisa sobre el documento. "Cría de perros: consejos sobre los alsacianos" es mucho más comprensible que "Perros, Alsacianos, Cría, Perro, Consejos, Gratuito, Mascota".

2.4. Añadir palabras clave y una descripción

Lo siguiente que se debe hacer puede parecer de entrada superfluo, ya que las personas que visiten vuestro sitio web no lo verán nunca: añadir una descripción y palabras clave. La una y las otras se deben añadir a `head` dentro de los elementos `meta`, tal como se puede ver en el ejemplo* siguiente de la web de Yahoo! Eurosport:

*** http://mosaic.uoc.edu/ac/le/operafiles/headwithmeta.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Yahoo! UK & Ireland Eurosport—Sports News | Live Scores | Sport</title>
```

```
    <meta name="description" content="Latest sports news and live scores from
    Yahoo! Eurosport UK. Complete sport coverage with Football results, Cricket
    scores, F1, Golf, Rugby, Tennis and more.">
    <meta name="keywords" content="eurosport,sports,sport,sports news,live
    scores,football,cricket,f1,golf,rugby,tennis,uk,yahoo">
</head>
<body>
</body>
</html>
```

Si abrís este documento en un navegador, no veréis nada de todo eso, pero si ponéis el documento en línea y los motores de búsqueda lo indexan, entonces el texto de esta descripción aparecerá bajo el vínculo en los resultados del motor de búsqueda, tal como se puede ver en la figura 2.

Figura 2

Las descripciones aparecen en las páginas de resultados de los motores de busca.

Ésta podría ser precisamente la información crucial que estaba buscando un posible visitante de vuestro sitio web y el motivo para hacer clic en el enlace. Las descripciones tienen otro uso más: algunos navegadores muestran la descripción como información adicional cuando se añade el documento a la lista de direcciones de interés, tal como se puede ver en la figura 3:

Figura 3

En algunos navegadores se ven las descripciones cuando se añade el documento a la lista de direcciones de interés.

Por lo tanto, aunque el hecho de añadir descripciones meta no tiene ninguna ventaja inmediata obvia, estas descripciones tienen una función muy importante para el éxito de vuestro documento. Eso mismo también sirve, aunque en un grado menor, para las palabras clave añadidas.

A pesar de que muchos años de abuso para falsear índices han hecho que los motores de búsqueda hayan dejado de tomarse seriamente las palabras clave, éstas aún pueden ser una buena herramienta que se puede utilizar para indexar rápidamente muchos documentos sin necesidad de tener que examinar y analizar su contenido. Podéis utilizar las palabras clave meta, por ejemplo, en un sistema de gestión de contenido escribiendo un script que las indexe y que haga que el motor de búsqueda funcione mucho más rápidamente. No cuesta nada ofrecer una manera de encontrar documentos sin necesidad de analizar su contenido. Si añadís algunas palabras clave en un elemento meta, tendréis la opción de realizar una búsqueda inteligente y rápida de vuestros sitios web en caso de que en el futuro los lleguéis a crear. Podéis imaginaros las palabras clave como unos pequeños puntos de libro que ponéis dentro de un libro muy grande para que os sea más fácil encontrar rápidamente una sección concreta sin tener que leer capítulos enteros.

2.5. ¿Y el aspecto? Añadir estilos

El otro elemento que podéis añadir al `head` de un documento son las normas para los estilos, que se definen en hojas de estilo en cascada (CSS, del inglés *cascading style sheets*). Los podéis integrar directamente en `head` utilizando un elemento `style`, como por ejemplo (`headinlinestyles.html*`):

*** http://mosaic.uoc.edu/ac/le/operafiles/headinlinestyles.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Breeding Dogs—Tips about Alsatians</title>
   <meta name="description" content="How to breed Alsatians, tips on proper
   breeding and information about common issues with this breed.">
   <meta name="keywords" content="Dogs,Alsatian,Breeding,Dog,Tips,Free,Pet">
   <style type="text/css">
      body{
         background:#000;
         color:#ccc;
         font-family: helvetica, arial, sans-serif;
      }
   </style>
</head>
<body>
<p>Test!</p>
```

```
</body>
</html>
```

Si abrís el archivo en un navegador, aparecerá la palabra *Test!* en gris sobre un fondo negro y el tipo de letra será Helvetica o Arial, según lo que haya instalado en vuestro sistema. El elemento `style` también puede contener otro atributo llamado `media`, que define qué tipo de soporte utilizará estos estilos; es decir, ¿queréis utilizar estos estilos cuando veis el documento en una pantalla de ordenador, en un dispositivo portátil o impreso? Podéis elegir entre varios tipos de soportes; los más útiles son:

- pantalla (*screen*): para la visualización en pantalla,
- impresión (*print*): define el aspecto que tendrá el documento impreso,
- aparatos portátiles (*handheld*): define el aspecto del documento en dispositivos móviles y otros dispositivos portátiles,
- presentaciones (*projection*): para presentaciones hechas en HTML.

Si, por ejemplo, queréis invalidar los colores utilizados en la pantalla y utilizar un tamaño mayor para el tipo de letra con el fin de mejorar la página para la impresión, podéis añadir otro bloque de estilos bajo el primero con un atributo `media` de `print`, como podéis ver a continuación (`headinlinestyles.html`*):

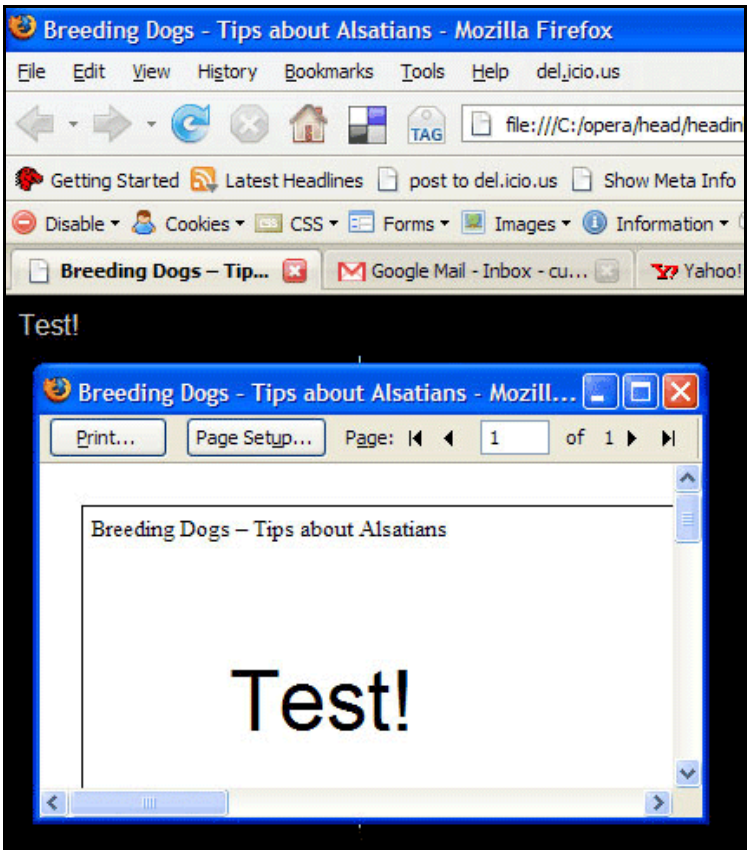
*** http://mosaic.uoc.edu/ac/le/operafiles/headinlinestylesprint.html**

```
<style type="text/css" media="print">
   body{
      background:#fff;
      color:#000;
      font-family: helvetica, arial, sans-serif;
      font-size:300%;
   }
</style>
```

Ahora, cuando imprimáis esta página web, el navegador sabrá que debe utilizar la hoja de estilos de impresión y no los estilos de pantalla. Podéis comprobarlo cargando `headinlinestylesmedia.html`* y seleccionando la previsualización de la impresión, como muestra la figura 4.

*** http://mosaic.uoc.edu/ac/le/operafiles/headinlinestylesprint.html**

Figura 4



Una hoja de estilos de pantalla y de impresión

Actualización

HTML4 y CSS2 usaban `media` exclusivamente con tipos de medios. Con HTML5 y CSS3 podemos extender esta funcionalidad a través de *media queries*. Una *media query* consiste en un medio, como los que hemos visto anteriormente, y, si lo deseamos, diferentes expresiones que comprueban características como la resolución de pantalla del navegador.

2.6. Añadir funciones dinámicas con JavaScript

Otro elemento que podéis añadir a `head` son *scripts* que ejecutará el navegador, y que por ello se conocen como "*scripts* de cliente", escritos en JavaScript. Como ya hemos visto, JavaScript añade un comportamiento dinámico al HTML estático, como por ejemplo efectos de animación, validación de datos de formularios u otras cosas que pasan cuando el usuario realiza determinadas acciones.

Podéis añadir JavaScript a un documento utilizando el elemento `script`. Cuando un navegador encuentra un elemento de este tipo, ignora todo lo demás y deja de analizar cualquier otra parte del documento mientras intenta ejecutar el código que hay en este elemento. Eso significa que si queréis estar seguros de que vuestro JavaScript estará disponible antes de que se cargue el documento principal, deberéis añadirlo a `head`. Por ejemplo, podéis avisar a la gente que os visite de que un enlace concreto los conducirá a otro servidor con el siguiente script (`headscript.html`*):

Hemos visto el JavaScript en el apartado 3 del módulo "Introducción al mundo de los estándares web".

* **http://mosaic.uoc.edu/ac/le/operafiles/headscript.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Breeding Dogs—Tips about Alsatians</title>
   <meta name="description" content="How to breed Alsatians, tips on proper breeding
   and information about common issues with this breed.">
   <meta name="keywords" content="Dogs,Alsatian,Breeding,Dog,Tips,Free,Pet">
   <style type="text/css" media="screen">
      body{
         background:#000;
         color:#ccc;
         font-family: helvetica, arial, sans-serif;
      }
      a {color:#fff}
   </style>
   <style type="text/css" media="print">
      body{
         background:#fff;
         color:#000;
         font-family: helvetica, arial, sans-serif;
         font-size:300%;
      }
   </style>
   <script>
      function leave(){
         return confirm("This will take you to another site,\n are you sure you want
         to go?")
      }
   </script>
</head>
<body>
Test!
<a href="http://dailypuppy.com" onclick="return leave()">The Daily Puppy</a>
</body>
</html>
```

Si abrís este ejemplo en vuestro navegador web y hacéis clic en el enlace, se os pedirá que confirméis la acción. Éste es sólo un ejemplo muy rápido de un script, pero queda muy lejos de las mejores prácticas actuales. Los otros tutoriales que encontraréis más adelante en esta misma asignatura explican las mejores prácticas de JavaScript y enseñan las técnicas de JavaScript con detalle, pero de momento no es necesario que os preocupéis de eso.

2.7. ¡Un momento! El uso de CSS integrado y JavaScript no es precisamente la mejor idea

Puede sonar muy fuerte, pero hay algo que debéis recordar siempre que construyáis sitios web: lo mejor que podéis hacer es reutilizar vuestro código tanto

como sea posible. Añadir estilos y scripts comunes a todo un sitio en todas y cada uno de las páginas no tiene demasiado sentido; más bien al contrario, ya que así es más difícil mantener un sitio completo y los documentos individuales quedan excesivamente cargados.

Es mucho mejor poner los estilos y scripts en archivos externos e importarlos hacia los archivos HTML cuando sea necesario; de esta manera, cuando se deban hacer cambios sólo habrá que actualizarlos en un lugar. Para vuestro JavaScript, eso es hace utilizando elementos script que no tienen ningún `script` en su interior, sino un enlace a un archivo externo utilizando un atributo `src`, tal como podéis ver en el código siguiente (`externaljs.html`*):

*** http://mosaic.uoc.edu/ac/le/operafiles/externaljs.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Breeding Dogs-Tips about Alsatians</title>
   <meta name="description" content="How to breed Alsatians, tips on proper
    breeding and information about common issues with this breed.">
   <meta name="keywords" content="Dogs,Alsatian,Breeding,Dog,Tips,Free,Pet">
   <style type="text/css" media="screen">
      body{
         background:#000;
         color:#ccc;
         font-family: helvetica, arial, sans-serif;
      }
      a {color:#fff}
   </style>
   <style type="text/css" media="print">
      body{
         background:#fff;
         color:#000;
         font-family: helvetica, arial, sans-serif;
         font-size:300%;
      }
   </style>
   <script src="leaving.js"></script>
</head>
<body>
Test!
<a href="http://dailypuppy.com" onclick="return leave()">The Daily Puppy</a>
</body>
</html>
```

Con CSS no es tan fácil. El elemento `style` no tiene ningún atributo `src`, con lo cual deberéis utilizar el elemento `link`; éste tiene un atributo `href` que especifica un fichero CSS externo para importarlo y un atributo `media` para definir si estos estilos se deben utilizar para pantalla, impresión, etc., de manera similar a lo que hemos visto antes. Poniendo tanto CSS como JavaScript en sus ficheros independientes se puede reducir de una manera muy importante la longitud del `head`, tal como se puede ver a continuación (`externalall.html`*):

*** http://mosaic.uoc.edu/ac/le/operafiles/externalall.html**

```
<!DOCTYPE html>
<html>
<head>
   <title>Breeding Dogs-Tips about Alsatians</title>
   <meta name="description" content="How to breed Alsatians, tips on proper
    breeding and information about common issues with this breed.">
   <meta name="keywords" content="Dogs,Alsatian,Breeding,Dog,Tips,Free,Pet">
   <link rel="stylesheet" type="text/css" media="screen" href="styles.css">
   <link rel="stylesheet" type="text/css" media="print" href="printstyles.css">
   <script src="leaving.js"></script>
</head>
<body>
Test!
<a href="http://dailypuppy.com" onclick="return leave()">The Daily Puppy</a>
</body>
</html>
```

Otras ventajas de mantener los estilos y scripts en sus ficheros independientes son:

- Todo será más rápido y fácil para los visitantes de vuestro sitio web; aunque se descarguen algunos archivos adicionales, la información sobre estilos y scripts no se repetirá en cada uno de los archivos de la página web (sólo se descargará una vez, en los archivos de estilos y scripts independientes), con lo cual, los archivos de página descargados serán más pequeños. Además, los archivos CSS y JavaScript se guardarán en la memoria caché (quedarán guardados temporalmente en vuestro equipo local) y la próxima vez que accedáis al sitio web los archivos ya estarán en el ordenador, lo que significa que no habrá que volver a cargarlos.

- La facilidad de mantenimiento. Los estilos y los scripts para todo el sitio web, que puede estar formado por miles de documentos, se encuentran en un único sitio; por lo tanto, si hay que cambiar algo, sólo lo tendréis que hacer en un archivo y no en miles.

Resumen

Esto es todo. Habéis aprendido las diferentes partes que puede haber en la sección de cabecera de los documentos HTML. Éstas son:

- `title`, que introduce el documento.
- Elementos `meta`, que contienen una descripción del contenido de este documento y palabras clave que permiten una indexación más sencilla del contenido.
- Elementos `link`, que indican los archivos CSS externos.
- Elementos `script`, que indican los archivos JavaScript externos.

Si todo es correcto, conseguiréis tener un documento rápido y fácil de encontrar y entender.

Preguntas de repaso

Como siempre, aquí tenéis unas cuantas preguntas de repaso para ver si habéis entendido todo lo que hemos explicado:

1. ¿Por qué es aconsejable añadir una descripción en un elemento `meta` aunque no se pueda ver en la pantalla?

2. ¿Qué ventaja tiene añadir JavaScript al `head` de un documento y no al `body`?

3. ¿Cómo podéis aprovechar el almacenaje en memoria caché de vuestro navegador y qué debéis hacer para sacarle partido?

4. Como los navegadores dan mucha importancia al título, ¿no sería útil llenarlo de palabras clave relacionadas? ¿Cuáles son las desventajas de esta práctica?

5. Como la pantalla del título puede ser un poco aburrida, ¿no sería bueno destacar en negrita algunas palabras con un elemento `b`? Es posible hacerlo?

3. Elegir el `doctype` correcto para los documentos HTML

Roger Johansson

El apartado anterior analizábamos la anatomía de la sección `head` de un documento HTML y explicábamos brevemente los diferentes elementos que puede haber en la cabecera y su función. En este apartado estudiaremos el elemento `doctype` con mucho más detalle: qué hace, cómo ayuda a validar el HTML, cómo elegir un `doctype` para el documento y la declaración XML, que no necesitaréis casi nunca pero con la que os encontraréis algunas veces.

3.1. En primer lugar, el `doctype`

Lo primero que debéis comprobar que haya en cualquier documento HTML que creéis es una declaración DTD. Si no habéis oído nunca hablar de la declaración DTD, no os preocupéis. Para facilitar las cosas, normalmente se conoce como `doctype`, que es el nombre que utilizaremos en este apartado.

Es muy posible que os estéis preguntando qué es una DTD o `doctype`. DTD son las siglas de *document type definition* (definición del tipo de documento), que, entre otras cosas, define los elementos y los atributos que se pueden utilizar en una versión concreta del HTML. Sí, no es ningún error, actualmente se utilizan en la Red diferentes versiones del HTML, pero esto no os debe preocupar en absoluto, ya que sólo os deberéis interesar por una.

El `doctype` se utiliza para dos cosas en diferentes tipos de software:

1. Los navegadores web lo utilizan para determinar el modo de representación que deben utilizar (más adelante ya hablaremos de los modos de representación).

2. Los validadores de etiquetado miran el `doctype` para determinar las normas con las que deben comprobar el documento (más adelante también hablaremos de ello).

Las dos os afectarán, pero de maneras diferentes que ya explicaremos más adelante en este mismo apartado.

Aquí tenéis un ejemplo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
            "http://www.w3.org/TR/html4/strict.dtd">
```

Es posible que no le encontréis ningún sentido, por lo que ahora explicaremos de una manera simplificada su construcción. Para una explicación con mucho más detalle de qué significa cada uno de los caracteres, podéis ver el artículo !DOCTYPE*.

*** http://www.blooberry.com/indexdot/html/tagpages/d/doctype.htm**

Las partes más importantes de `doctype` son las dos cadenas delimitadas por comillas. "`-//W3C//DTD HTML 4.01//EN`" indica que éste es un documento DTD publicado por el W3C, que la DTD describe la versión 4.01 del HTML y que el idioma utilizado en la DTD es el inglés.

La segunda cadena, "`http://www.w3.org/TR/html4/strict.dtd`", es una URL que indica el documento DTD utilizado para este `doctype`.

Aunque un `doctype` puede tener un aspecto un tanto extraño, las especificaciones HTML y XHTML lo exigen. Si no incluís ninguno, aparecerá un error de validación cuando comprobéis la sintaxis de vuestro documento con el validador de etiquetado del W3C u otras herramientas que comprueben los documentos HTML por si contienen errores. Algunos navegadores incluso contienen esta función por defecto, mientras que otros pueden incluirla si se instala una extensión.

En cualquier caso, si usáis el `doctype` de HTML5, que es el que hemos visto hasta ahora en todos los ejemplos, la cosa queda muy simplificada, puesto que este es el más sencillo posible:

```
<!DOCTYPE html>
```

3.2. Cambios de `doctype` y modos de representación

Si no proporcionáis ningún `doctype`, los navegadores seguirán gestionando y mostrando el documento de todos modos; deben intentar mostrar todo tipo de cosas extrañas que se pueden encontrar en la Red, por lo que no pueden ser muy quisquillosos. No obstante, sin un `doctype` los resultados no siempre serán los esperados a causa de una cosa conocida como "doctype sniffing" o "doctype switching".

La mayoría de los navegadores del siglo XXI miran el `doctype` de todos los documentos HTML que encuentran y lo utilizan para decidir si el autor de los documentos se ha preocupado de escribir adecuadamente el HTML y el CSS según los estándares de la Red.

Si encuentran un `doctype` que indica que el documento está bien codificado, utilizan un modo conocido como "modo de estándares" para maquetar la página. En el modo de estándares, los navegadores intentan normalmente mostrar la página según las especificaciones de CSS; se fían de que la persona que ha creado el documento sabía qué estaba haciendo.

Por otra parte, si encuentran un `doctype` anticuado o incompleto, utilizan el modo Quirks (*peculiaridades*), que es más compatible con las prácticas y los navegadores antiguos. El modo Quirks presupone que el documento es antiguo o que no se ha creado teniendo en cuenta los estándares de la web; la página web se seguirá mostrando, pero se necesitará más potencia de procesamiento para hacerlo y es muy probable que el resultado sea algo extraño, nada parecido a lo que se esperaba.

Las diferencias están básicamente relacionadas con la manera de mostrar el CSS y sólo en unos pocos casos con la manera de tratar el HTML en sí. Como diseñadores o desarrolladores de webs, obtendréis los mejores resultados asegurándoos de que todos los navegadores utilicen su modo de representación de estándares, y por ello deberíais ceñiros a los estándares de la web y utilizar un `doctype` adecuado.

3.3. Validación

Como ya hemos comentado, los validadores también utilizan el `doctype`; ya lo comentaremos con más detalle más adelante en esta serie de apartados. De momento, todo lo que debéis saber es que se utiliza un validador para comprobar que la sintaxis de vuestros documentos HTML sea correcta y que no contenga ningún error. Los programas validadores miran el `doctype` utilizado para determinar las normas que se deben utilizar. Es algo similar a indicar a un corrector ortográfico el idioma en el que está escrito un documento. Si no lo sabe, no sabrá qué normas ortográficas y de gramática debe utilizar.

3.4. Elegir un `doctype`

Así pues, ahora que ya sabéis que hay que introducir un `doctype` y para qué se utiliza, ¿cómo podéis saber cuál hay que utilizar? De hecho, no hay uno, sino que hay muchos. Incluso podéis crear uno propio, siempre y cuando tengáis el nivel de conocimientos necesario para hacerlo. Aquí, sin embargo, no hablaremos de muchos `doctypes` diferentes. Haremos las cosas fáciles y sólo mencionaremos dos.

Si vuestro documento es HTML5, utilizad el ya visto:

```
<!DOCTYPE html>
```

Si por algún motivo vuestro documento debe utilizarse en navegadores incompatibles con HTML5 (en general esto pasará porque hay una base importante de usuarios que aún usan versiones antiguas de IE), utilizad el siguiente para indicar que estáis usando HTML 4.01:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
   "http://www.w3.org/TR/html4/strict.dtd">
```

Si vuestro documento es XHTML, utilizad el siguiente:

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

Notas

- El motivo por el que los `doctype` de HTML 4.01 y XHTML 1 necesitaban especificar una DTD y el HTML5 no es que los dos primeros se basaban en SGML, y HTML5 no. El significado de SGML va más allá de los objetivos de este curso.

- El XHTML "real" se debe comunicar al navegador como XML, pero los detalles sobre cómo y cuándo hacerlo y las implicaciones que tiene, quedan fuera del alcance de este apartado en concreto.

Estos `doctypes` garantizarán que los navegadores utilicen su modo de estándares a la hora de trabajar con vuestro documento. El efecto más evidente que tendrá sobre vuestro trabajo es que obtendréis unos resultados más uniformes a la hora de diseñar el documento con CSS.

3.5. La declaración XML

Nota

Esta sección sólo es relevante si debéis usar HTML 4.01 o XHTML 1.0.

Ya hemos señalado antes que el `doctype` debe ser la primera cosa que haya dentro de vuestros documentos HTML. Pues bien, eso es en realidad una versión simplificada de la verdad. También hay que tener en cuenta la declaración XML.

Es posible que en algunos documentos XHTML hayáis visto un fragmento de código con un aspecto similar al siguiente antes del `doctype`:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Esto se conoce como declaración XML, y cuando está presente debe estar insertada antes del `doctype`.

La versión 6 de Internet Explorer tiene un problema con esta declaración: hace que pase al modo Quirks y, tal como ya hemos explicado anteriormente, no es muy probable que queráis que eso suceda.

Por suerte, la declaración XML no es necesaria si no es que enviáis vuestros documentos XHTML como XML a los navegadores (podéis ver la nota al margen

sobre el XHTML) y utilizáis una codificación de caracteres diferente de UTF-8 y vuestro servidor no envía un encabezamiento HTTP que determina la codificación de caracteres.

Las probabilidades de que sucedan todas estas cosas al mismo tiempo son muy pequeñas, por lo que la manera más sencilla de solucionar el problema de Internet Explorer es sencillamente omitir la declaración XML. ¡Pero no os olvidéis del `doctype`!

Resumen

Hay que incluir siempre uno de los `doctypes` mencionados como primer elemento dentro de todos vuestros documentos HTML. Esto garantizará que los validadores sepan la versión del HTML que utilizáis y así podrán informar correctamente de cualquier error que hayáis cometido. También garantizará que todos los navegadores más nuevos utilizarán su modo de estándares, con lo que obtendréis unos resultados más uniformes cuando diseñéis el documento con el CSS.

Preguntas de repaso

Después de leer este apartado, deberíais poder responder a las siguientes preguntas:

1. ¿Cuáles son los dos objetivos principales de incluir un `doctype` en los documentos HTML?

2. ¿Cuáles son las ventajas de utilizar un `doctype` estricto en lugar de uno transicional?

3. ¿Por qué es problemática la declaración XML?

4. Un `doctype` que no hemos mencionado en este apartado es el de definición de marcos; intentad descubrir qué hace y por qué no se debe utilizar.

Lecturas complementarias

“Don't forget to add a doctype”
http://www.w3.org/QA/Tips/Doctype

“Recommended DTDs to use in your Web document”
http://www.w3.org/QA/2002/04/valid-dtd-list.html

“Fix Your Site With the Right DOCTYPE!”
http://www.alistapart.com/articles/doctype/

"Activating Browser Modes with Doctype"
http://hsivonen.iki.fi/doctype/

"The Opera 9 DOCTYPE switches"
http://www.opera.com/docs/specs/doctype/

"Quirks mode and strict mode"
http://www.quirksmode.org/css/quirksmode.html

"Transitional vs. StrictMarkup"
http://24ways.org/2005/transitional-vs-strict-markup