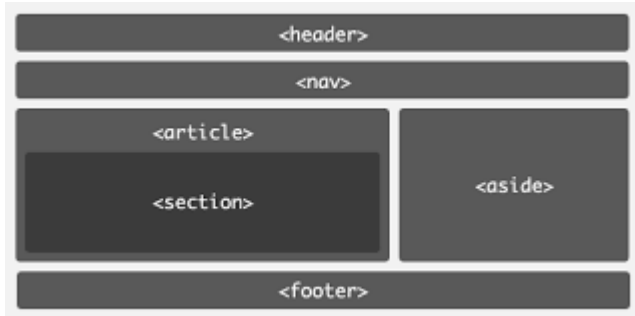


Lógica y Estructura de Datos

HTML5



Técnico Superior en Desarrollo de Software

Esc. Superior N° 49 "Cap. Gral. J.J. Urquiza"

Año 2017

Ing. Álvaro Hergenreder

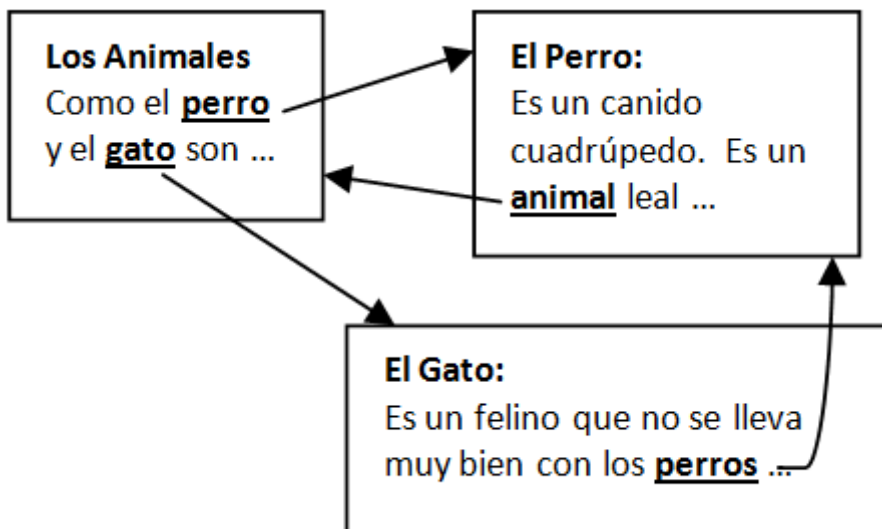
World Wide Web, www o La Web

Esta es una pequeña introducción a lo que es la World Wide Web empezando por la definición más sencilla:

World Wide Web = Hipertexto + Internet

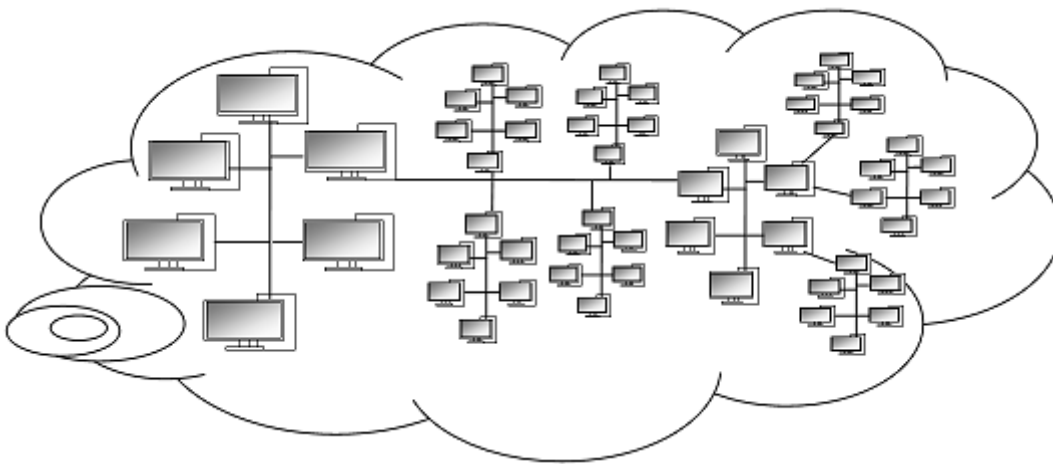
Hipertexto

Hipertexto son documentos de texto entrelazados. El enlace normalmente es una palabra o frase subrayada que apunta a otro documento. Al hacer click sobre el enlace el documento actual es reemplazado por el indicado en el enlace.



Internet

Internet es el sistema mundial de interconexión de redes de ordenadores basado en el protocolo de comunicaciones IP. Algunos servicios existentes sobre esta red son: correo (e-mail), transferencia de archivos (ftp) y el servicio de documentos de hipertexto llamado World Wide Web (www) basado en el protocolo http.



World Wide Web

La World Wide Web fue inventada en los 1990s por Tim Berners-Lee en CERN y es el concepto de Hipertexto aplicado a Internet. Cada documento llamado "Página web" se encuentra en un servidor y es visto usando un programa al que Berners-Lee llamó Browser.

Browser o Cliente HTTP

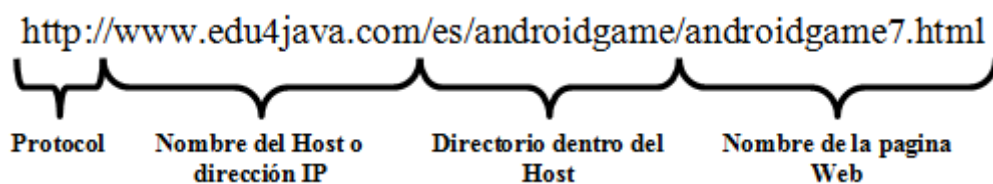


Página web

Una página web está escrita en lenguaje HTML y tiene un nombre o identificador único en todo Internet. Este identificador es llamado URL.

URL "Uniform Resource Locator"

Un URL está compuesto de:



Sitio Web

Un Sitio Web es un conjunto de páginas webs que normalmente se encuentran en el

mismo Host. Por ejemplo `www.terciariourquiza.edu.ar` es un sitio web.

Browser, Navegador, Explorador o Cliente HTTP

Un Browser es el programa que nos permite navegar por la World Wide Web. Si lo alimentamos con la dirección de una página Web URL se encargará de traerla desde el otro lado del mundo si es necesario.

Cliente HTTP

El Browser trabaja conjuntamente con un servidor HTTP que es el programa que almacena las páginas Web. Esta relación entre los dos programas es conocida como "Cliente - Servidor" donde el Browser es el cliente. La forma en que se comunican se llama "Protocolo HTTP" de allí el nombre "Cliente HTTP".

Browser o Cliente HTTP



Nota: *la mayoría de los clientes http permiten leer archivos directamente desde un sistema de archivos (disco duro, CD, etc.) sin necesidad de un servidor HTTP. En principio vamos a trabajar de esta forma, editando el código html y ejecutando en la PC en modo local.*

HTML

La página Web que obtiene el Browser normalmente esta en lenguaje HTML. Este HTML contiene el texto que veremos en pantalla e indicaciones de cómo presentar este texto. Por ejemplo se puede indicar que parte del texto es un título o que parte va a ser un enlace hacia otra página Web. El Browser interpreta el HTML y presenta en la pantalla el texto formateado según estas indicaciones.

JavaScript

Además de indicaciones de cómo formatear el texto la página HTML puede contener código JavaScript. Esto significa que la página puede contener un programa que

se ejecutará en la máquina del cliente. Inicialmente estos scripts solo se usaban para pequeños efectos visuales, hoy día se usan intensamente, a través de JQuery y bajo un nuevo concepto llamado AJAX.

Nota: el sitio <http://maps.google.com/> es quizás el ejemplo de AJAX más famoso.

Plugins o complementos

Además de soportar JavaScript los navegadores suelen permitir la instalación de programas complementos llamados plugins. Estos plugins permiten extender las capacidades estándares. Algunos permiten la corrección ortográfica, descargar archivos de forma más eficiente, etc.

Flash

Seguramente que hoy día el complemento más importante es "Adobe flash player" que permite ver animaciones y videos embebidos en las páginas Web.

Applet Java

Esta es una tecnología muy poco usada actualmente que permitía embeber un programa java en una página Web.

En resumen

Todas estas tecnologías son conocidas como tecnologías del lado del cliente. Más adelante veremos tecnologías del lado del servidor.

Servidor Web o Servidor HTTP

Habíamos visto que el protocolo HTTP de la Web funciona bajo el paradigma "Cliente - Servidor". El cliente es el Browser que solicita páginas Web y el servidor HTTP o servidor Web las entrega. En este apartado vamos a centrarnos en las tecnologías asociadas al servidor conocidas como **"tecnologías del lado del servidor"**.

Nota: Un servidor Web no está limitado a entregar solo páginas Web. Puede entregar cualquier tipo de archivo como pdfs, planillas de cálculo o incluso un formato inventado por nosotros.

Browser o Cliente HTTP



Sitio Web Estático o Dinámico

En un comienzo los servidores guardaban las páginas Web como archivos en unidades de almacenamiento. Un servidor Web se limitaba a buscar el archivo solicitado y enviarlo al cliente. Este tipo de sitios Web son llamados Estáticos.

Con el tiempo se comenzó a utilizar programas en el servidor que generaban las páginas Web en el momento en que eran solicitadas. Hoy día es muy común que las páginas de un sitio Web sean generadas con programas escritos en PHP o Java. Este tipo de sitios Web son llamados Dinámicos.

De libros digitales a programas en la red

Con los sitios dinámicos muchos sitios pasaron de ser libros digitales a programas en la red. Por ejemplo, si ustedes tienen una cuenta de correo electrónico en gmail de Google la pueden manejar a través del sitio Web <http://www.gmail.com/>. Pueden leer nuevos correos, borrar o crear nuevos correos y enviarlos.

Programación Web

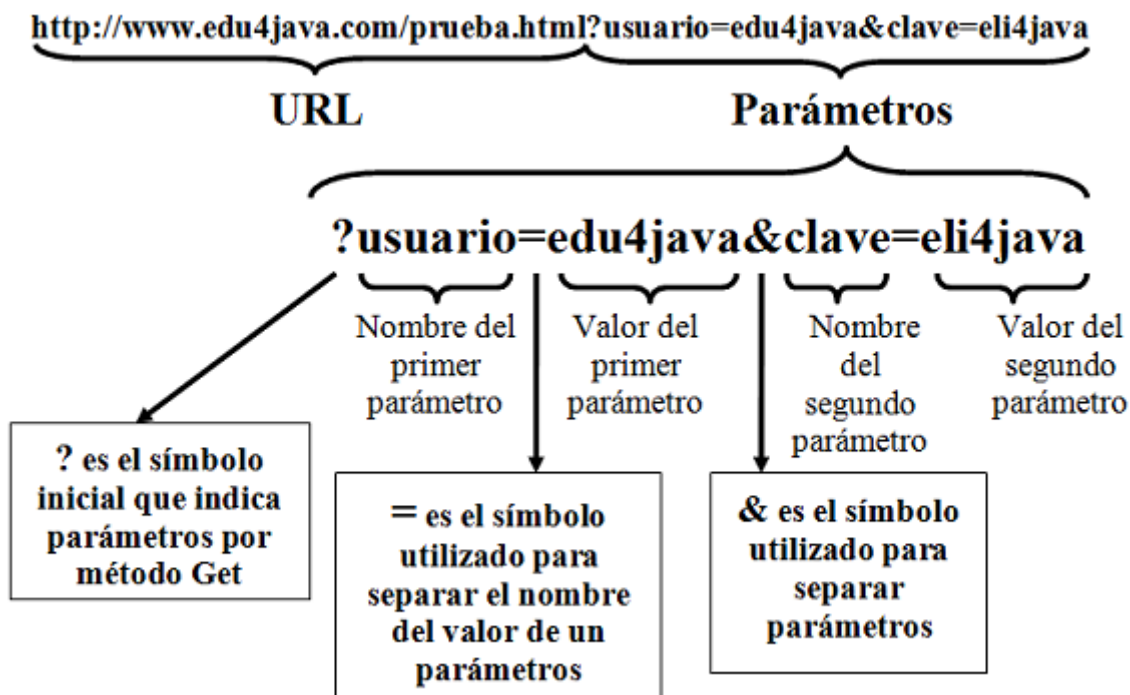
Estos Programas Web tuvieron tanto éxito que la mayor cantidad de programación en estos últimos años ha sido mayoritariamente "Programación Web". Bancos, gobierno, tiendas, agencias de viajes y muchos más tienen sitios Web que en realidad son programas donde nos piden un usuario y una contraseña y entonces podemos acceder a nuestros datos y a múltiples servicios.

Parámetros y formularios

Los programas Web rompieron con la idea clásica de documentos enlazados de hipertexto. Surge la necesidad de enviar datos desde las páginas Web al servidor (por ejemplo el usuario y la contraseña). Cuando esto sucedió tanto el HTTP como el HTML fueron modificados para soportar el envío de parámetros y formularios desde el cliente al servidor. Para esta comunicación se inventaron dos métodos llamados Get y Post.

Método Get

Esta técnica consiste en agregar al final del URL un signo "?" y pares de "nombre" y "valor" separados por un símbolo "&". Ejemplo:



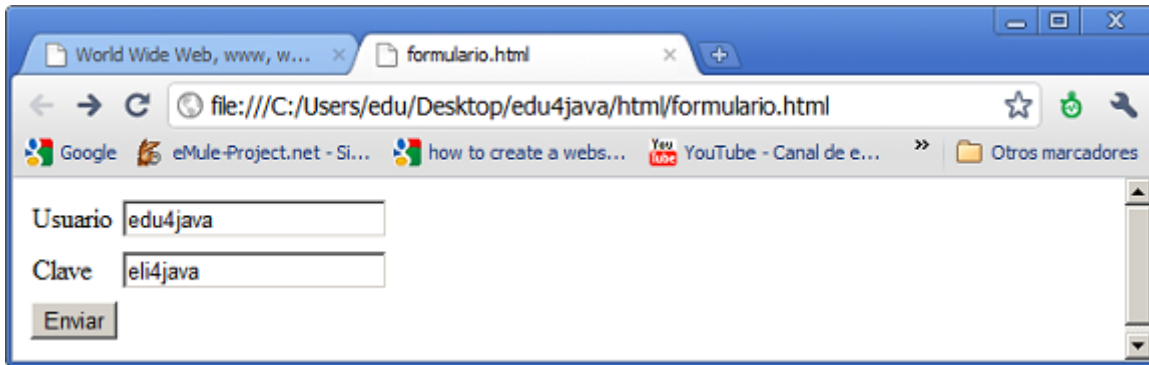
El método Get tiene dos principales inconvenientes. El primero es que se ven los datos en el URL. Esto es muy malo en el ejemplo de usuario y clave ya que cualquiera que viera la casilla del URL en el navegador vería nuestra clave. La segunda desventaja es que el largo del URL está limitado por lo que esto nos limita la cantidad de caracteres que podemos enviar.

Método Post

Este método igual que el Get envía al servidor pares de "nombre" y "valor" pero lo hace dentro del mensaje HTTP que solicita una página al servidor. De esta forma los datos no son visibles en el URL y no existe la limitación dada por el largo máximo de un URL. El método Post está asociado con el de formulario HTML.

Formulario HTML

A continuación se ve un simple formulario Web en el Browser. Cuando oprimamos el botón Enviar, el contenido de las cajas de texto rellenas con edu4java y eli4java serán enviadas al servidor.

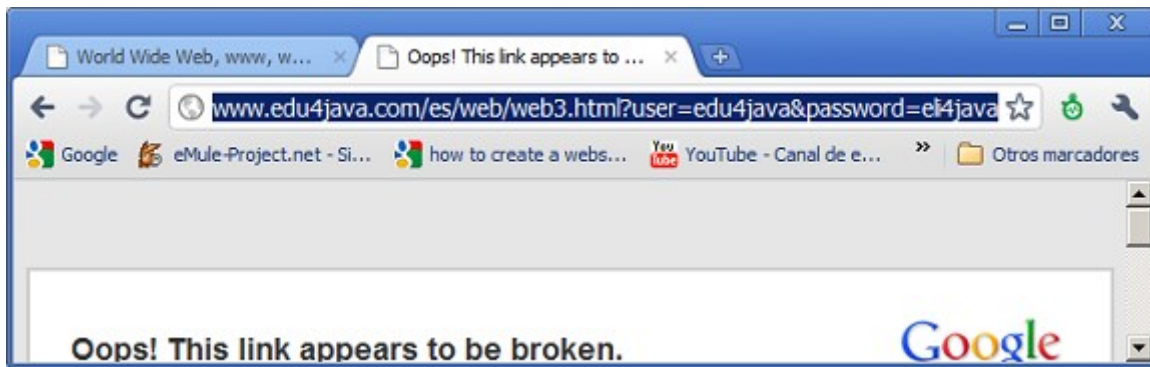


Este es el código HTML para generarlo:

```
<html>
<body>
    <form action="http://www.edu4java.com/es/web/web3.html" method="get">
        <table>
            <tr>
                <td>User</td>
                <td><input name="user" />
            </td>
            </tr>
            <tr>
                <td>password</td>
                <td><input name="password" /></td>
            </tr>
        </table>
        <input type="submit" />
    </form>
</body>
</html>
```

- Podemos ver que la etiqueta `form` tiene un atributo `action` que indica a que servidor se enviará la petición de página.
- Las etiquetas `input` representan las cajas de texto y el atributo `name` indica el nombre del parámetro usado para enviar la información de su de caja de texto al servidor.
- El atributo `method` indica el método Get del que hablamos antes para enviar la información.

Después de oprimir enviar podemos ver en el URL de la página resultado como fueron enviados los datos del formulario al servidor.

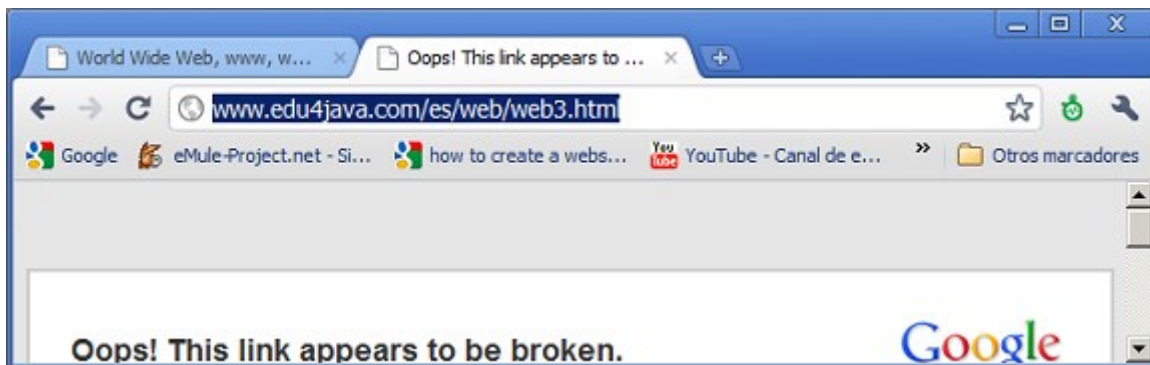


Nota: ignorar el error "Oops! This link ..." esto es porque www.edu4java.com es una página estática y no está preparada para recibir parámetros.

Si cambiamos el método de Get a Post

```
<form action="http://www.edu4java.com/es/web/web3.html" method="post">
```

el servidor recibiría la información pero no se vería en el URL



PHP y Java del lado del servidor: Servlet y Jsp

Los lenguajes PHP y Java se han convertido en los más usados del lado del servidor. PHP es visto por muchos como la solución más económica para proyectos simples, aunque con él se han desarrollado aplicaciones como Facebook. Java es hoy día el lenguaje más usado en el planeta y es el más usado a nivel universitario, empresarial y gubernamental. Mucha gente cree que la elección de java para un gran proyecto es obligatoria. La comunidad java posee innumerable proyectos y Frameworks comerciales y de código abierto como Struts, Spring, Maven, Google Web Toolkit GWT, etc.

HTML. Introducción

Lo único que falta saber para poder estar en la web es aprender el lenguaje de la web: HTML.

HTML, siglas de HyperText Markup Language («lenguaje de marcas de hipertexto»),

hace referencia al lenguaje de marcado para la elaboración de páginas web.

Es un estándar que sirve de referencia para la elaboración de páginas web en sus diferentes versiones, define una estructura básica y un código (denominado código HTML) para la definición de contenido de una página web, como texto, imágenes, etc.

Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

Para la escritura de este lenguaje, se crean **etiquetas** que aparecen especificadas a través de corchetes o paréntesis angulares: < y >. Entre sus componentes, los **elementos** dan forma a la estructura esencial del lenguaje, ya que tienen dos propiedades (el contenido en sí mismo y sus atributos).

La función del navegador es leer los documentos HTML y convertirlos en páginas visibles y/o audibles. El navegador no muestra las etiquetas HTML, utiliza las etiquetas para interpretar el contenido de la página.

Los elementos HTML forman los ladrillos de todas las páginas web. HTML nos permite insertar imágenes y objetos y crear formularios interactivos. Nos da una plataforma para crear documentos estructurados, suministrándonos estructuras semánticas para texto como cabeceras, párrafos, listas, links, etc.

Por otra parte, cabe destacar que el HTML permite ciertos códigos que se conocen como scripts, los cuales brindan instrucciones específicas a los navegadores que se encargan de procesar el lenguaje. Entre los scripts que pueden agregarse, los más conocidos y utilizados son JavaScript y PHP.

Los navegadores web también pueden consultar las Hoja de estilos en cascada (CSS) para definir el diseño del texto y otro material. W3C, que son los encargados de los estándares de HTML y CSS, animan el uso de CSS para la presentación de los documentos en HTML.

Tags, Elementos y Estructura de un documento HTML

A continuación vemos un ejemplo de código fuente de una página Web o documento HTML.

```
<html>
  <head>
    <title>Programacion I</title>
  </head>
  <body>
    <h1>Hola Alumnos</h1>
    <p>Bienvenidos a nuestra primera <br> pagina Web. <br/> </p>
  </body>
```

`</html>`

Tag HTML. ¿Que es un Tag HTML?

Cada palabra rodeada por los símbolos "<" y ">" en el código fuente anterior es llamada Tag. Existen 3 tipos de Tags:

1. Tag de inicio o apertura. Ejemplo: `<h1>`
2. Tag de fin o cierre. Ejemplo: `</h1>`
3. Tag de inicio y cierre. Ejemplo: `
`

Elemento HTML

Un tag de inicio y su correspondiente tag de fin componen un elemento HTML.

Ej: `<title>Hola Mundo</title>`

Lo que está entre un tag de inicio y su correspondiente tag de fin es el contenido del tag.

Un tag de inicio y cierre por si solo compone un elemento HTML. Ej: `
`

Siempre es recomendable utilizar tags de apertura y cierre en cada elemento html.

Estructura básica de un documento HTML

El primer tag HTML de una página Web debería ser `<html>` y el último `</html>`. En otras palabras toda la página Web debe estar contenida en un elemento HTML.

Dentro del elemento HTML existen el elemento **head** que contendrá información de configuración y el elemento **body** que contendrá todo lo que es visible de nuestra página.

Cuando un Browser interpreta este código fuente anterior el resultado es:



En este ejemplo podemos ver como el elemento **title** define el título de la página que aparece en la pestaña superior.

Dentro del elemento **body** existe un elemento **h1** que define un encabezado principal y un elemento **p** que define un párrafo. Dentro del elemento **p** hay un elemento **br** que define un salto de carro.

Es decir, dentro de **body** no solo está la información a ser representada en la página Web sino que existen marcadores o tags que indican como representar la información.



Elementos HTML ordenados por función

A continuación tenemos una lista de los elementos HTML ordenados en base a su función. Esto nos ayudará a saber que elemento utilizar en cada caso para poder facilitar el desarrollo de páginas web compatibles con distintos navegadores y plataformas.

Lista de elementos HTML ordenados en base a su función:

** Etiqueta añadida en HTML5.*

Básico:

Etiqueta	Descripción
<!--...-->	Define un comentario. Cualquier información que se ponga dentro de las etiquetas <!-- y --> dentro del cuerpo de la página, será

	ignorado por el navegador.
<!DOCTYPE>	Define el tipo de documento
<html>	Define la raíz de un documento HTML
<title>	Define el título de un documento
<body>	Este elemento es utilizado para definir el cuerpo del documento.
<h1> -<h6>	Define encabezados en HTML
<p>	Especifica un párrafo
 	Define un salto de línea.
<hr>	Define un cambio temático en el contenido

Secciones/Estilo

Etiqueta	Descripción
<style>	Este elemento define el estilo del documento
<div>	Define una sección en un documento.
	Agrupar varios elementos en línea seguidos dentro de un mismo bloque, para después darles formato con la hoja de estilo.
<header>*	Define un encabezado para un documento o sección
<footer>*	Define un footer para un documento o sección.
<section>*	Define una sección en un documento
<article>*	Define una composición independiente dentro de un documento, página, aplicación o web, que puede ser distribuida independientemente o reutilizada.
<aside>*	Define contenido secundario al contenido de la página.
<details>*	Define detalles adicionales que el usuario puede ver o esconder.
<dialog>*	Define una caja de diálogo o ventana.
<summary>*	Define un encabezado para el elemento <details>

Información Meta

Etiqueta	Descripción
<head>	Define información sobre el documento
<meta>	Define metadata de un documento HTML
<base>	Define cuál es la URL absoluta para todos aquellos enlaces que sean relativos en la página
<basefont>	Especifica un color, tamaño y fuente por defecto, para todo el texto del documento. No soportado en HTML5. Declarado en desuso en HTML 4.01

Formato

Etiqueta	Descripción
<acronym>	Permite a los autores indicar claramente una secuencia de caracteres que componen un acrónimo o abreviación de una palabra. No soportado en HTML5.
<abbr>	Define una abreviación
<address>	Define información de contacto del propietario del documento
	Este elemento indica que el texto englobado tiene que ser mostrado en negrita.
<bdi>*	Aisla una parte del texto que puede tener un formato diferente del texto principal.
<bdo>	Este elemento es utilizado para sobrecribir la dirección del texto.
<big>	Este elemento define texto grande. No soportado en HTML5.
<blockquote>	Este elemento se utiliza para designar una cita de texto larga. Normalmente, se puede reconocer porque está indentado.
<center>	Este elemento hace que el texto englobado esté centrado dentro de los márgenes. No soportado en HTML5. Declarado en desuso en HTML 4.01.
<cite>	Especifica una cita o referencia a otro lugar
<code>	Indica que el texto englobado es código de programación.
	Define texto que ha sido eliminado de un documento.
<dfn>	Define un término de una definición.
	Define énfasis en un texto.
	Define fuente, color y tamaño para un texto. No soportado en HTML5.
<i>	Este elemento se usa para representar un texto con un estilo de fuente en itálica
<ins>	Define el texto que ha sido insertado dentro del documento
<kbd>	Este elemento se utiliza para avisar al usuario que tiene que introducir datos por teclado
<mark>*	Define un texto resaltado en el documento
<meter>*	Define una medida escalar dentro de un rango determinado
<pre>	Define un texto con formato previo
<progress>*	Representa el progreso de una tarea
<q>	Define una cita de texto corta(en una línea)
<rp>*	Define que mostrar en los navegadores que no soportan anotaciones ruby
<rt>*	Este elemento define la pronunciación de caracteres (para tipografía del este asiático)

<ruby>*	Define una anotación ruby (para tipografía del este asiático)
<s>	Esta etiqueta define texto con un estilo de fuente tachado
<samp>	Muestra un mensaje de un programa o script
<small>	Se utiliza para presentar un texto con un estilo de fuente pequeña
<strike>	Este elemento se utiliza para representar un texto con un estilo de fuente tachado. No soportado en HTML5. Declarado en desuso en HTML 4.01.
	Se utiliza para representar texto importante
<sub>	Representa un subíndice
<time>*	Define la fecha y hora
<tt>	Define texto con un estilo de fuente en teletipo. No soportado en HTML5.
<u>	Define texto con un estilo de fuente subrayado.
<var>	Define una variable
<wbr>*	Define donde un texto se puede romper para cambios de línea

Formularios

Etiqueta	Descripción
<form>	Define un formulario HTML para datos del usuario.
<input>	Este elemento se utiliza para ingresar datos por parte del usuario
<textarea>	Este elemento especifica una área para escribir texto
<button>	Se utiliza para crear botones de un formulario
<select>	Muestra varias opciones al usuario y permite que elija entre ellas una o más en una lista desplegable
<optgroup>	Especifica un grupo de opciones en una lista desplegable.
<option>	Este elemento define una opción en una lista desplegable.
<label>	Define una etiqueta para un elemento <input>
<fieldset>	Agrupar elementos relacionados en un formulario.
<legend>	Este elemento define un título para un elemento <fieldset>, <figure> o <details>
<datalist>*	Especifica una lista de opciones predefinidas para una caja de texto input
<keygen>*	Define un campo generador de claves en formularios
<output>*	Especifica el resultado de un cálculo

Tablas

Etiqueta	Descripción
----------	-------------

<table>	Define una tabla
<caption>	Define el título de una tabla
<th>	Representa la celda del encabezado de una tabla
<tr>	Define una fila dentro de una tabla
<td>	Define una celda de una tabla
<thead>	Agrupar el contenido del encabezado de una tabla
<tbody>	Agrupar el contenido del body de una tabla
<tfoot>	Agrupar el contenido del footer de una tabla
<col>	Define una columna dentro de una tabla y se utiliza para agrupar y alinear.
<colgroup>	Especifica un grupo de una o más columnas de una tabla para darles formato.

Listas

Etiqueta	Descripción
	Define una lista no ordenada
	Este elemento especifica una lista ordenada
	Define un ítem de una lista
<dir>	Define una lista directorio. No soportado en HTML5. Declarado en desuso en HTML 4.01.
<dl>	Define una lista de definición.
<dt>	Define un término (un ítem) dentro de una lista de definición.
<dd>	Define una descripción de un término en una lista descriptiva
<menu>	Define un menú
<command>*	Define un botón comando al que el usuario puede llamar.

Links

Etiqueta	Descripción
<a>	Principalmente utilizado como hipervínculo.
<link>	Este elemento define la relación entre el documento y un recurso externo (normalmente utilizado con hojas de estilo)
<nav>*	Define links de navegación

Images

Etiqueta	Descripción
	Define una imagen

<map>	Define un mapa de imagen en el lado del cliente
<area>	Esta etiqueta define un área dentro de un mapa de imagen
<canvas>*	Utilizada para dibujar gráficos, mediante scripts. (normalmente JavaScript)
<figcaption>*	Define un título para un elemento <figure>
<figure>*	Especifica contenido independiente.

Audio/Video

Etiqueta	Descripción
<audio>*	Se utiliza para representar contenido de audio en los documentos. Añadido en HTML5, puede contener diferentes fuentes de audio, representadas utilizando el atributo "src" o el elemento <source>.
<source>*	Define las fuentes para los elementos multimedia como video y audio.
<track>*	Define determinadas características para las pistas de texto de elementos multimedia como; vídeos o audios.
<video>*	Define un video o película

Frames

Etiqueta	Descripción
<frame>	Define una ventana dentro de un frameset. No soportado en HTML5.
<frameset>	Define un conjunto de "frames". No soportado en HTML5.
<noframes>	Especifica el contenido alternativo para los navegadores que no soportan frames. No soportado por HTML5.
<iframe>	Define un documento HTML dentro de otro documento HTML

Programación

Etiqueta	Descripción
<script>	Define un script dentro del documento
<noscript>	Especifica un contenido alternativo para navegadores que no soportan scripts.
<applet>	Esta etiqueta define un applet embebido dentro del documento. No soportado en HTML5. Declarado en desuso en HTML 4.01.
<embed>*	Se utiliza para declarar un contenedor para una aplicación externa (no- HTML)

<object>	Define un objeto embebido
<param>	Este elemento especifica un parámetro para un objeto

HTML5

Es la quinta revisión importante del lenguaje HTML. Su principal objetivo es mejorar el lenguaje añadiendo soporte para las últimas tecnologías multimedia, así como manteniendo el código legible al humano y entendible para la máquina y dispositivos (navegadores, etc.).

HTML5 ofrece nuevas características (elementos, atributos, manejadores de eventos, y APIs) para desarrollos de páginas web más simples y para formas más complejas de manejo. La especificación HTML5 se basa en HTML 4.01 Strict, pero a diferencia de especificaciones anteriores, HTML5 no utiliza una Definición de Tipo de Documento (DTD).

En su lugar, utiliza como base, el Modelo de Objetos del Documento (DOM, el "árbol" creado por la estructura del documento), en vez de un conjunto determinado de reglas sintácticas.

Se diferencia también de las especificaciones anteriores en que se incluyen instrucciones detalladas de como los navegadores deben utilizar el HTML mal especificado.

Escribir HTML5

Se puede escribir páginas en HTML5 utilizando el mismo software que se utiliza para escribir páginas HTML.

Podemos utilizar un editor de texto sencillo como el Notepad (en Windows) o como el TextEdit (en Mac). Varias herramientas actuales de diseño (como el Adobe Dreamweaver y el Microsoft Expression Web) tienen plantillas que te permiten crear rápidamente nuevos documentos HTML5.

En realidad, la estructura básica de una página HTML5 es tan sencilla que se puede utilizar cualquier editor web, para crearla, incluso si el editor web no está diseñado para HTML5.

Compatibilidad HTML5

¿Cuáles son los navegadores que soportan HTML5? HTML5 es una colección de estándares independientes.

Algunos ya son compatibles; y otros no lo serán hasta dentro de algunos años (o quizá nunca lo sean).

El resto está en tierras intermedias, lo que significa que HTML5 funciona en

algunas versiones de algunos navegadores.

A continuación tenemos algunos de los navegadores que soportan una gran parte de HTML sin necesitar muchos rodeos:

- Internet Explorer 9 y versiones superiores
- Firefox 3.5 y versiones superiores
- Google Chrome 8 y versiones superiores
- Safari 4 y versiones superiores
- Opera 10.5 y versiones superiores

La compatibilidad aumenta con las versiones superiores. Firefox 5, por ejemplo, soporta mejor HTML que Firefox 3.5.

Ejemplo de estructura global con HTML5

Los documentos HTML siguen una estricta organización.

Cada parte del documento está diferenciada, declarada y englobada con etiquetas específicas. En esta sección vamos a ver cómo construir la estructura global de un documento HTML con los nuevos elementos incorporados en HTML5.

Doctype

En primer lugar, tenemos que indicar el tipo de documento que estamos creando. En HTML5 esto es muy sencillo:

```
<!DOCTYPE html>
```

Esta línea deberá ser la primera línea del documento, sin espacios ni líneas previas. Esto es una manera de activar el modo estándar y forzar a los navegadores a interpretar el HTML5 siempre que sea posible, o a ignorarlo cuando no lo sea.

```
<html>
```

Después de declarar el tipo de documento, se construye la estructura del árbol con HTML. El elemento raíz del árbol es el elemento `<html>`. Este elemento engloba todo el código HTML.

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

El atributo *lang* en la etiqueta de inicio `<html>` es el único atributo que debemos especificar en HTML5. Este atributo define el lenguaje del contenido del documento que estamos creando- en este caso, *es* para español.

HTML5 es muy flexible con respecto a la estructura y los elementos utilizados en su desarrollo. El elemento `<html>` puede ser incluido sin atributos o no ser incluido.

Sin embargo, por compatibilidad y otras razones, se recomienda seguir unas reglas básicas. Aquí vamos a aprender cómo construir un documento HTML siguiendo un manual de buenas prácticas.

`<head>`

Sigamos con la construcción de nuestra plantilla. El código HTML incluido en las etiquetas `<html>` tiene que estar dividido en dos secciones principales. Como ya ocurría en versiones anteriores de HTML, la primera sección es el "head" y la segunda el "body". El siguiente paso, por tanto, es crear dos secciones en el código, utilizando esos dos elementos: `<head>` y `<body>`.

El elemento `<head>` va en primer lugar, y como el resto de los elementos estructurales, tiene un tag de apertura y otro de cierre.

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

El "tag" no ha cambiado con respecto a otras versiones y su función es la misma. Dentro de los "tags" `<head>`, definiremos el título de nuestra página, declararemos la codificación de caracteres, incluiremos cualquier información general que queramos sobre el documento, e incorporaremos archivos externos con los estilos, scripts o incluso imágenes necesarias para mostrar la página.

A excepción del título y algunos iconos, el resto de la información incluida en el documento dentro de las etiquetas `<head>` no será visible.

`<body>`

La siguiente sección que forma parte de la organización principal del documento HTML es el "body". Este "cuerpo" es la parte visible del documento y viene especificado con la etiqueta `<body>`.

Esta etiqueta no ha cambiado con respecto a las versiones anteriores de HTML:

```
<!DOCTYPE html>
<html lang="es">
<head>
```

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

<meta>

A continuación vamos a desarrollar la cabecera del documento. Existen algunas modificaciones e innovaciones dentro de esta sección, y una de ellas es la etiqueta que define la codificación de los caracteres del documento. Es la etiqueta *meta* y especifica cómo se debe presentar el texto dentro de la pantalla.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
<meta charset="utf-8">  
</head>  
<body>  
</body>  
</html>
```

La modificación para este elemento de HTML5, es como en muchos casos, una simplificación. La etiqueta *meta* nueva para la codificación de caracteres es más corta y más simple. Igualmente, se puede cambiar utf-8 por la codificación que se prefiera, y se pueden añadir otras etiquetas meta como "description" o "keywords", tal y como se ve en el ejemplo que tenemos a continuación:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
<meta charset="utf-8">  
<meta name="description" content="Esto es un ejemplo de HTML5">  
<meta name="keywords" content="HTML5">  
</head>  
<body>  
</body>  
</html>
```

Nota: Como vemos, la etiqueta <meta> especifica el tipo y el atributo *content* declara su valor, pero ninguno de estos valores serán mostrados en la pantalla. En HTML5, no es necesario poner una barra de cierre, pero se recomienda por razones de compatibilidad.

<title>

La etiqueta <title>, como siempre, especifica el título del documento, y no hay nada nuevo en HTML5.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Esto es un ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>Título del documento</title>
</head>
<body>
</body>
</html>
```

Nota: Normalmente, este texto es mostrado por los navegadores en la parte superior de la ventana.

<link>

Otro elemento importante que se incluye dentro de la cabecera es <link>. Este elemento se utiliza para incorporar estilos, scripts, imágenes o iconos de archivos externos. Uno de los usos más comunes de <link>, es el de insertar un archivo CSS externo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Esto es un ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>Título del documento</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
</body>
</html>
```

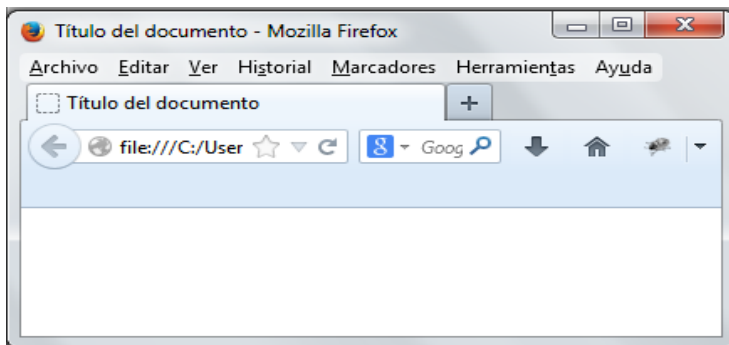
En HTML5, ya no hay que especificar el tipo de hoja de estilo que estamos insertando, por lo que el atributo *type* ha sido eliminado.

Solamente necesitamos dos atributos para incluir nuestro archivo de estilos: *rel* y *href*. El atributo *rel* se refiere a la relación entre el documento y el archivo que estamos incluyendo. En este caso tiene el valor= "stylesheet", lo que le indica al navegador que el archivo edu4javastyles.css es un archivo CSS con estilos necesarios para mostrar la página.

Nota: Una hoja de estilos es un conjunto de reglas de formato que nos va a ayudar a cambiar la presentación de nuestro documento –por ejemplo, el tamaño o el color del texto–.

Sin estas reglas, el texto y otros elementos se mostrarán en la pantalla utilizando los estilos estándares suministrados por los navegadores (tamaño y colores por defecto).

Nuestro código se vería así:



Estructura del cuerpo– HTML5

La estructura del cuerpo (el código entre los tags <body>) generará la parte visible de nuestro documento. Será el código que genere nuestra página web.

HTML ofrece diferentes formas para construir y organizar la información en el cuerpo del documento.

Uno de los primeros elementos que ofrece HTML para este propósito es <table>. Las tablas, aunque no fueran ideadas para eso, permiten organizar data, texto, imágenes dentro de filas y columnas.

Con el tiempo y gradualmente, otros elementos han reemplazado la función de las tablas, aportando diferentes maneras de hacer lo mismo con menos código y de una manera más rápida, facilitando la creación, portabilidad y mantenimiento.

El elemento <div> empezó a extenderse y a dominar el campo. Sin embargo, el elemento <div>, como el <table>, no suministran mucha información sobre las partes del cuerpo que están representando. Cualquier cosa puede ir dentro de las etiquetas de apertura y cierre <div>, desde imágenes, menus, textos, links, scripts, formularios, etc.

Para los usuarios, saber que hay dentro de las etiquetas no es importante, pero

para los navegadores es crucial saberlo para hacer la correcta interpretación de los documentos.

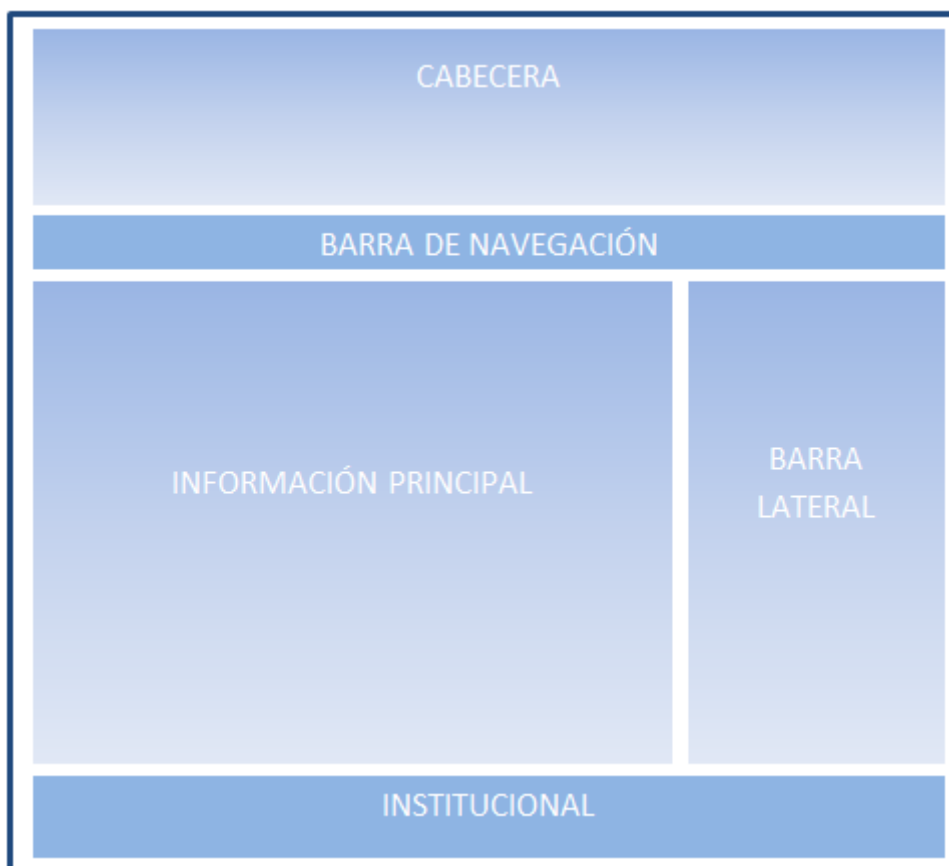
Teniendo esto en cuenta, HTML5 ha incorporado nuevos elementos que ayudan a identificar cada parte del documento y a organizar el cuerpo.

Como utilizar estos nuevos elementos, es cosa nuestra, pero las palabras clave seleccionadas para cada uno, nos ayudará a saber cuál es su función.

Normalmente una página web o aplicación web está dividida en diferentes áreas visuales para poder mejorar la experiencia del usuario y su interactividad. Las palabras clave que representan cada elemento nuevo de HTML5, están relacionadas a estas áreas visuales.

Organización

Cada diseñador crea sus propios diseños, pero en general podremos identificar dentro de cada página web las siguientes secciones:



Arriba de todo, tenemos la **Cabecera**, que es el lugar donde solemos tener el logo, nombre, y pequeñas descripciones de nuestra página web.

Debajo de la cabecera, podemos ver la **Barra de Navegación**, en la que casi cualquier desarrollador ofrece un menú o lista de links para poder navegar por el sitio web.

Los usuarios son llevados desde esta barra a diferentes páginas o documento,

normalmente en el mismo sitio web.

El contenido más importante de la página se localiza normalmente en el medio de la distribución. Esta sección muestra la información importante y links. Casi siempre está dividida en varias filas y columnas.

La sección de **Información Principal**, podría tener por ejemplo, una lista de artículos, descripciones de productos, entradas de blog y cualquier otra información importante.

La **Barra Lateral** podría tener por ejemplo, una lista de links apuntando a cada uno de estos ítems.

Abajo del todo, tenemos una barra llamada **Barra Institucional**. Se llama así porque es el área de la página donde tenemos información general del sitio web, el autor o la compañía, además de links con referencia a términos y condiciones, mapas e información adicional que el desarrollador cree importante compartir.

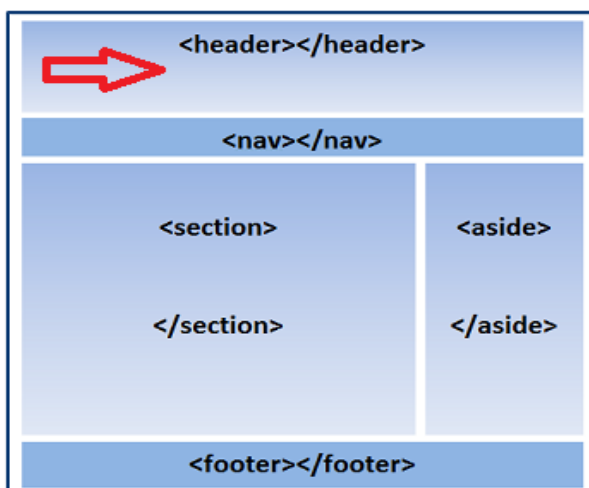
La barra institucional es el complemento a la Cabecera, y es considerada como una parte esencial de la estructura de la página.

Estructura del cuerpo en HTML5. Etiqueta `<header>`

HTML5 tiene una estructura y un diseño básicos y aporta nuevos elementos para diferenciarlos y declararlos.

Con ello podemos informar al navegador, la función de cada sección.

A continuación tenemos una representación visual de la organización de las secciones utilizando las etiquetas HTML5. En esta sección vamos a aprender sobre la primera de estas etiquetas: `<header>`



<header>

Uno de los nuevos elementos incorporados en HTML5 es el <header>. No debe ser confundido con la etiqueta <head>, que hemos utilizado anteriormente para definir el título de nuestra página.

De la misma manera que la etiqueta <head>, la etiqueta <header> aporta información introductoria (como títulos, subtítulos, logos...), pero en otro ámbito.

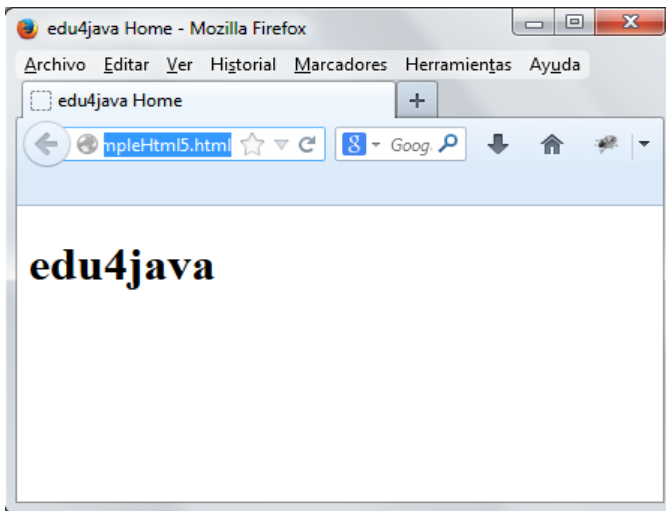
Mientras que la etiqueta <head> tiene la función de suministrar información sobre todo el documento, la etiqueta <header> aporta información del cuerpo del documento o de alguna sección dentro del cuerpo.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
</body>
</html>
```

En el código, vemos que definimos el título de la página web, utilizando la etiqueta <header>. Recuerde que este título no es el mismo que el título general del documento definido previamente en el <head>.

La inserción del elemento <header> representa el principio del cuerpo y de la parte visible del documento.

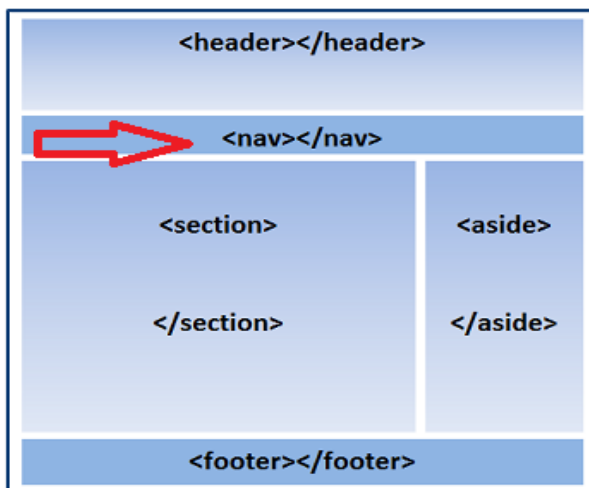
A partir de ahora vamos a poder ver los resultados de nuestro código en la ventana de nuestro navegador:



Nota: El elemento `<h1>` es un elemento de HTML utilizado para definir un titular. El número indica la importancia del titular y su contenido. El elemento `<h1>` es el de mayor importancia y `<h6>` es el de menor, por lo que el `<h1>` será utilizado para mostrar el título principal y el resto para títulos secundarios.

Estructura del cuerpo en HTML5. Etiqueta `<nav>`

En esta sección vamos a aprender sobre la etiqueta `<nav>`



`<nav>`

La siguiente sección que vamos a explicar es la **Barra de Navegación**. Esta barra es generada en HTML5 con la etiqueta `<nav>`.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
```

```
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
</body>
</html>
```

Como vemos en el código anterior, el elemento `<nav>` se encuentra dentro de las etiquetas `<body>`, pero después de las etiquetas de cierre de header (`</header>`), no entre las etiquetas `<header>`. Esto es porque el `<nav>` no forma parte del header, sino que es una nueva sección.

HTML5 es muy versátil y nos suministra los parámetros y elementos básicos con los que podemos trabajar. Como utilizar esos elementos para crear nuestra estructura y el orden que escojamos es una decisión nuestra.

Un ejemplo de esta versatilidad es la etiqueta `<nav>`, que podemos insertar o bien dentro del elemento `<header>` o bien dentro de cualquier otra sección del cuerpo. Igualmente, hay que tener en cuenta, que estas etiquetas nuevas se han creado para proveer a los navegadores de más información y así ayudar a cualquier programa nuevo o dispositivo del mercado a identificar las partes más relevantes del documento.

Para mantener nuestro código portable y legible, recomendamos seguir los estándares y mantenerlo lo más ordenado posible. El elemento `<nav>` está hecho para incluir ayudas a la navegación como el menu principal o los bloques principales de navegación y se debería utilizar para este propósito.

Nota: En este ejemplo, vamos a hacer una lista con las opciones de menu de nuestra página web. Entre las etiquetas `<nav>`, existen dos elementos que se utilizan para crear una lista. Utilizamos el elemento `` para definir la lista. Dentro de estas etiquetas `` vemos varias etiquetas `` con diferente texto, que representan las opciones de menu.

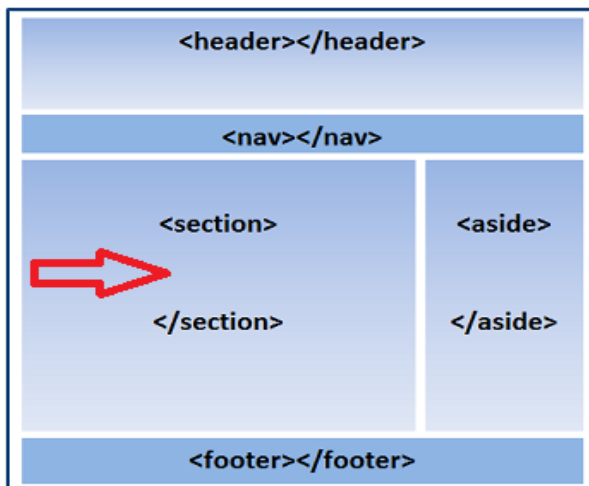
Estas etiquetas ``, se utilizan para definir cada ítem de la lista.

Este es el resultado de nuestro código:



Estructura del cuerpo en HTML5. Etiqueta `<section>`

En esta sección vamos a aprender sobre la etiqueta `<section>`



`<section>`

Siguiendo nuestro diseño estándar tenemos la **Barra de Contenido Principal** que contiene la información más relevante del documento que puede disponerse, por ejemplo, en varios bloques o columnas.

Dado que el propósito de estas columnas o bloques es más general, el elemento

HTML5 que lo especifica se llama simplemente `<section>`.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
TUTORIALES JAVA, HTML, JUEGOS ANDROID, SQL...
</section>
</body>
</html>
```

Igual que la Barra de Navegación, la Barra de Contenido Principal es una sección separada del resto. Por lo tanto esta sección se localiza debajo de la etiqueta de `</nav>`.

IMPORTANTE: Las etiquetas que representan cada sección del documento aparecen listadas dentro del código, una encima de la otra, pero en las páginas web algunas de estas secciones aparecerán una al lado de la otra. (Por ejemplo las columnas de Contenido Principal y la Barra Lateral).

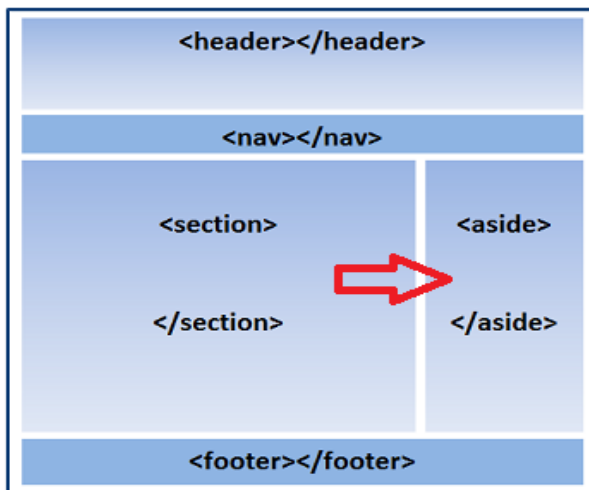
En HTML5, la presentación en pantalla se delega a CSS. El diseño se llevará a cabo asignando un estilo CSS a cada elemento.

Este es el resultado hasta ahora:



Estructura del cuerpo en HTML5. Etiqueta `<aside>`

En esta sección vamos a aprender sobre la etiqueta `<aside>`



`<aside>`

En una distribución estándar de una página web, la barra lateral está situada al lado de la barra de contenido principal.

Es una columna o sección que normalmente contiene datos relacionados al contenido principal pero con una relevancia o importancia menor.

En el diseño estándar de un blog, la barra lateral contiene una lista de links. La información dentro de esta barra está relacionada con el contenido principal de la página pero no es relevante por si sola.

Siguiendo con el ejemplo del blog, podemos decir que las entradas del blog son

importantes, pero los links y resúmenes de esas entradas son sólo una ayuda navegacional y no en lo que el lector o usuario está más interesado.

En HTML5, podemos diferenciar este tipo de información secundaria con el elemento `<aside>`.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
TUTORIALES JAVA, HTML, JUEGOS ANDROID, SQL...
</section>
<aside >
English Version
</aside>
</body>
</html>
```

El elemento `<aside>` puede estar localizado en cualquier parte del diseño; la etiqueta no define la posición, como vemos en el ejemplo. El elemento `<aside>` sólo describe la información que engloba, no un lugar en la estructura y se puede utilizar siempre que el contenido no quiera ser considerado como el contenido principal del documento.

Podemos utilizar el elemento `<aside>`, por ejemplo, dentro de un elemento `<section>`, o incluso dentro de una sección con información relevante, en una cita dentro de un texto.

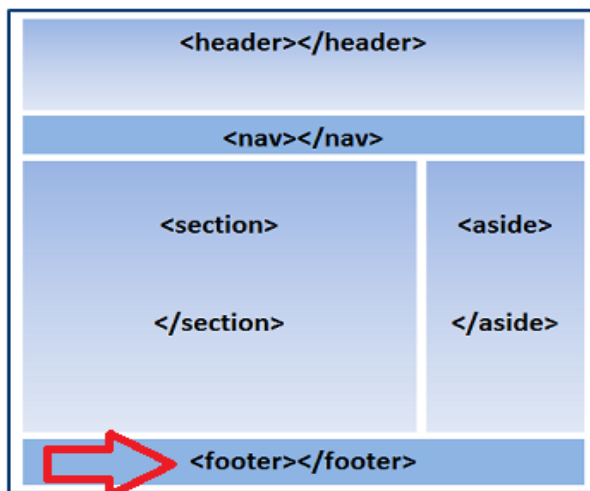
Hay que recordar que en HTML5, la presentación en pantalla de estos elementos ha sido delegada a CSS. El diseño se consigue asignando estilos CSS a cada elemento.

Este es el resultado de nuestro código sin CSS:



Estructura del cuerpo en HTML5. Etiqueta `<footer>`

En esta sección vamos a aprender sobre la etiqueta `<footer>`



`<footer>`

Para dar por concluido nuestra plantilla o estructura básica de nuestro documento HTML5, sólo necesitamos un elemento más. Ya tenemos el **header** del cuerpo, secciones con **ayudas navegacionales**, el **contenido principal** y la **barra**

lateral con información adicional.

Lo único que necesitamos para cerrar el diseño es darle una terminación al cuerpo del documento. HTML5 tiene un elemento específico para este fin llamado **<footer>**

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
TUTORIALES JAVA, HTML, JUEGOS ANDROID, SQL...
</section>
<aside >
English Version
</aside>
<footer>
Copyright &copy; 2012–2013
</footer>
</body>
</html>
```

En el diseño estándar de una página web, la sección llamada **Barra Institucional** será definida con etiquetas **<footer>**. Se llama así porque la barra representa el fin (o pie) de nuestro documento y esta parte de la página web es utilizada normalmente para compartir información general sobre el autor o la compañía que ha creado el proyecto; como copyright, términos y condiciones, etc.

Normalmente, el elemento **<footer>** representa el fin del cuerpo de nuestro

documento y tiene el propósito antes descrito. Sin embargo, la etiqueta <footer> puede ser usada varias veces dentro del cuerpo para representar el fin de las diferentes secciones. (La etiqueta <header> también puede ser utilizada varias veces dentro del cuerpo).

Este es el resultado de nuestro código sin CSS:



Contenido del cuerpo. Etiqueta article

El cuerpo de nuestro documento ya está completo. La estructura básica de la página web está terminada y ahora tenemos que trabajar en el contenido. Los elementos HTML5 que hemos visto hasta ahora nos ayudan a identificar cada sección del diseño y a asignar una función intrínseca para cada uno de ellos, pero lo que es realmente importante es lo que hay dentro de cada una de esas secciones.

La mayoría de los elementos ya estudiados fueron creados para dar una estructura a la página web que los navegadores y nuevos dispositivos puedan identificar.

Las etiquetas <body> son para declarar el cuerpo o las partes visibles del documento, las etiquetas <header> son para englobar información importante para el cuerpo, las etiquetas <nav> para aportar ayuda a la navegación, las etiquetas <section> contienen el contenido más importante y las etiquetas <aside> y <footer> suministran información adicional.

Pero ninguno de estos elementos declaran nada sobre el contenido en sí mismo.

Todos tienen una función estructural muy específica.

Cuanto más profundizamos en el documento, más nos acercamos a la definición del contenido. Esta información estará compuesta de elementos visuales diferentes como títulos, textos, imágenes, videos y aplicaciones interactivas entre otras. Necesitamos poder diferenciar estos elementos y establecer relaciones entre ellos.

<article>

De la misma manera en que los blogs están divididos en entradas, las páginas webs están divididas en partes que tienen características similares. El elemento <article> nos permite identificar cada una de estas partes.

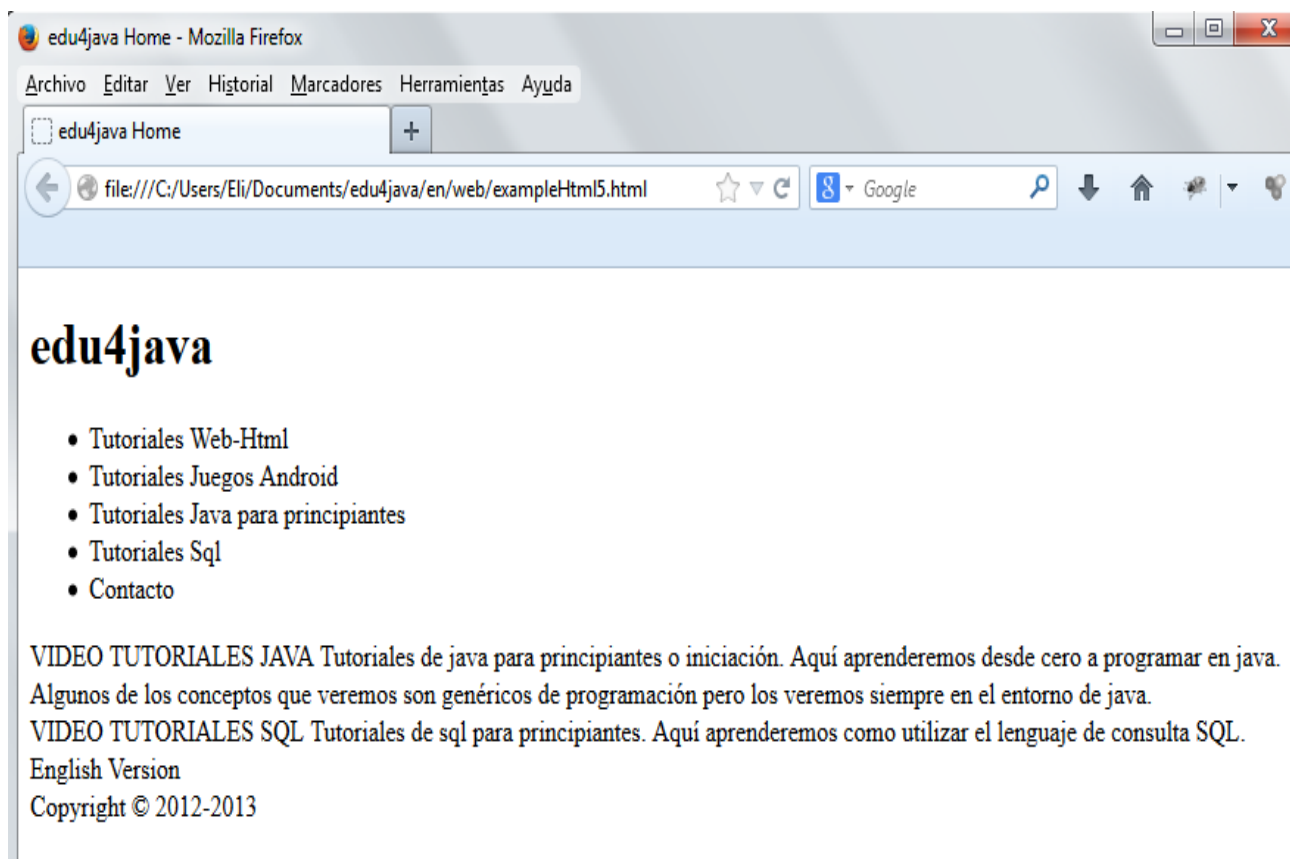
```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
<article>
VIDEO TUTORIALES JAVA
Tutoriales de java para principiantes o iniciación. Aquí aprenderemos desde
cero a programar en java. Algunos de los conceptos que veremos son genéricos de
programación pero los veremos siempre en el entorno de java.
</article>
<article>
VIDEO TUTORIALES SQL
```

Tutoriales de sql para principiantes. Aquí aprenderemos como utilizar el lenguaje de consulta SQL.

```
</article>
</section>
<aside >
English Version
</aside>
<footer>
Copyright &copy; 2012-2013
</footer>
</body>
</html>
```

Como podemos ver en el código, las etiquetas `<article>` están dentro de las etiquetas `<section>`. Las etiquetas `<article>` pertenecen a esa sección. Son sus hijos, de la misma manera que las etiquetas que están dentro de las etiquetas `<body>` son hijos del cuerpo. Pero como con todo hijo del nodo cuerpo, las etiquetas `<article>` son colocadas una después de la otra, ya que cada una de ellas es una parte independiente de `<section>`.

Este es el resultado:



Como hemos dicho antes, la estructura HTML puede ser descrita como un árbol, con el elemento `<html>` de raíz. Otra forma de describir las relaciones entre los

elementos es denominarlos, padres, hijos o hermanos según la posición en la estructura del árbol. Por ejemplo, en un documento HTML típico, el elemento `<body>` es el hijo del elemento `<html>` y hermano del elemento `<head>`. Tanto, `<body>` como `<head>`, tiene el elemento `<html>` como padre.

El elemento `<article>` no está limitado por su nombre –no sólo habla de artículos–.

El elemento `<article>` está creado para contener una unidad independiente de contenido, por lo que puede llevar un post de un foro, un artículo de una revista, una entrada de un blog, un comentario de un usuario, etc. Este elemento agrupará unidades de información relacionadas entre sí, sin tener en cuenta la naturaleza de la información.

Como parte independiente del documento, el contenido de cada elemento `<article>` tendrá su estructura independiente. Para definir esta estructura podemos sacar ventaja a la versatilidad de las etiquetas `<header>` y `<footer>` ya estudiadas anteriormente. Estas etiquetas son portables y pueden ser utilizadas fuera del cuerpo, en todas las secciones del documento.

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
<article>
<header>
<h1>VIDEO TUTORIALES JAVA </h1>
```

</header>

Tutoriales de java para principiantes o iniciación. Aquí aprenderemos desde cero a programar en java. Algunos de los conceptos que veremos son genéricos de programación pero los veremos siempre en el entorno de java.

<footer>

<p>comments (0)</p>

</footer>

</article>

<article>

<header>

<h1>VIDEO TUTORIALES SQL </h1>

</header>

Tutoriales de sql para principiantes. Aquí aprenderemos como utilizar el lenguaje de consulta SQL.

<footer>

<p>comments (0)</p>

</footer>

</article>

</section>

<aside >

English Version

</aside>

<footer>

Copyright © 2012-2013

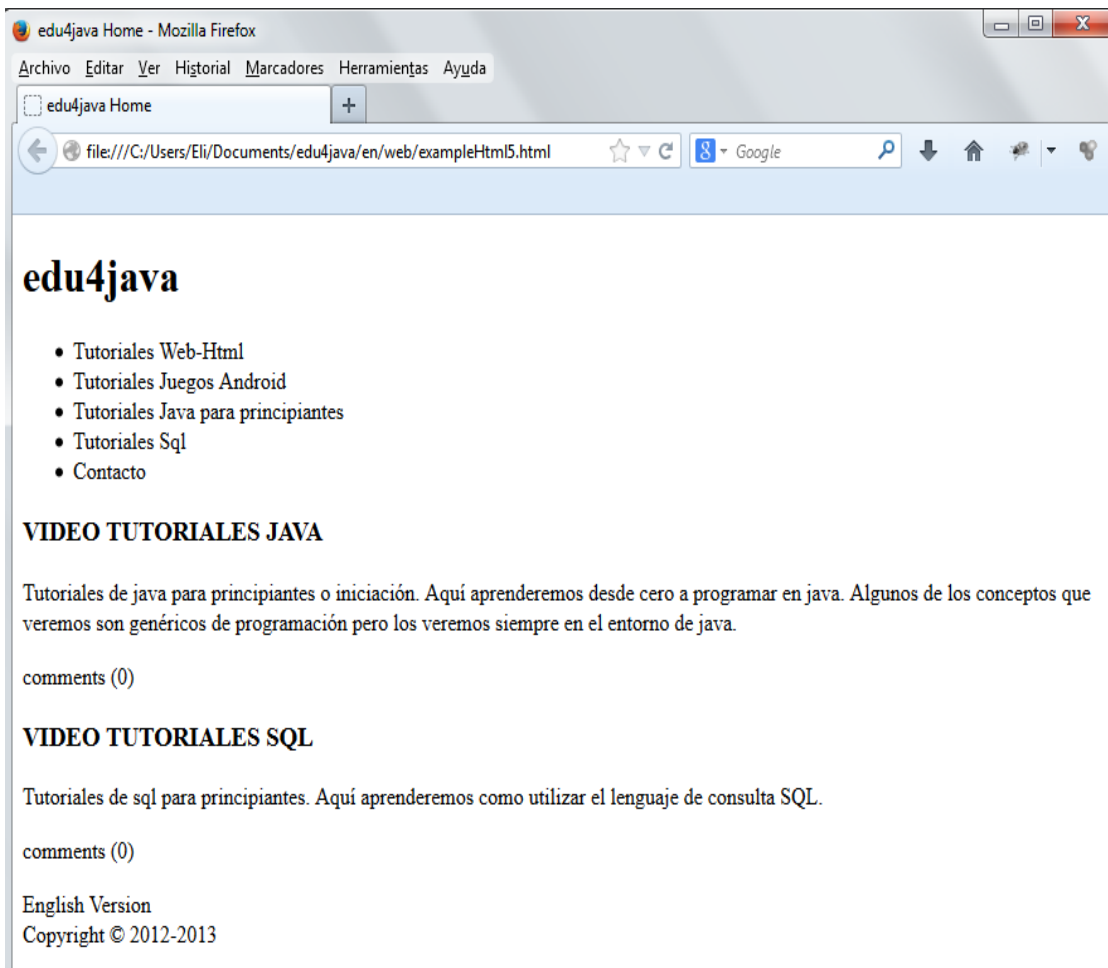
</footer>

</body>

</html>

Aquí tenemos un ejemplo de una unidad de información creada con el elemento <article> con una estructura propia. Primero con la etiqueta <header> que contiene el título definido con el elemento <h1>. A continuación tenemos el contenido propiamente dicho, un texto describiendo el tutorial. Y finalmente, después del texto, tenemos la etiqueta <footer> especificando el número de comentarios.

Este es el resultado del código:



Contenido del cuerpo. Etiqueta hgroup

Anteriormente ya hemos incorporado etiquetas `<h1>` para especificar el título, dentro de un elemento `<header>`, al principio del cuerpo o al principio de cada `<article>`.

Las etiquetas `<h1>` son necesarias para crear el encabezado de cada parte del documento. A veces necesitamos también añadir subtítulos o más información para describir el contenido de la página web o de alguna sección. De hecho, el elemento `<header>` está hecho para incluir otros elementos— como por ejemplo, una tabla de contenidos, formularios de búsqueda, logos o textos cortos.

Para construir el header, podemos utilizar las ventajas que nos dan el resto de las etiquetas H: `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` y `<h6>`. Pero para propósitos de procesamiento interno, y para evitar generar múltiples secciones o subsecciones durante la interpretación del documento, estas etiquetas se deben agrupar. Para esto el HTML5, nos proporciona el elemento `<hgroup>`.

<hgroup>

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
<li>Contacto</li>
</ul>
</nav>
<section>
<article>
<header>
<hgroup>
<h1>VIDEO TUTORIALES JAVA </h1>
<h2>Primer programa, con Eclipse en Español. Como programar en java </h2>
</hgroup>
</header>
Esta sección es el primero de una serie de tutoriales en los que vamos a
aprender los conceptos básicos del lenguaje orientado a objetos JAVA.
<footer>
<p>comments (0)</p>
</footer>
</article>
</section>
<aside >
English Version
</aside>
<footer>
Copyright &copy; 2012-2013
</footer>
```

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

Las etiquetas H deben guardar su jerarquía, lo que significa que primero se declara el título con la etiqueta <h1>, y después se utiliza el <h2> para el subtítulo, etc. Sin embargo, al contrario de versiones más antiguas de HTML, HTML5 nos permite reutilizar las etiquetas H y construir esta jerarquía las veces que se quiera en cada sección del documento.

El elemento <hgroup> es necesario cuando tenemos un título y subtítulo o más de una etiqueta H junta en el mismo <header>. Este elemento sólo puede contener etiquetas H. Si solamente tenemos una etiqueta <h1>, no hace falta agruparlo. Por ejemplo en el <header> del cuerpo no hemos utilizado este elemento ya que sólo tenemos un elemento H en el interior. Hay que recordar que el elemento <hgroup> está hecho para agrupar etiquetas H, tal y como indica su nombre.

Este es el resultado:



Los navegadores y programas que ejecutan páginas web, leen el código HTML y

crean su propia estructura interna para interpretar y procesar cada elemento. Esta estructura interna se divide en secciones que no tienen relación con las divisiones en el diseño o en el elemento `<section>`. Estas son secciones conceptuales generadas durante la interpretación del código.

El elemento `<header>` no crea solamente una de estas secciones conceptuales, esto significa que los elementos dentro del `<header>` representarán diferentes niveles y pueden generar internamente diferentes secciones. El elemento `<hgroup>` se creó con el propósito de agrupar las etiquetas H y así evitar que el navegador hiciera malas interpretaciones.

Contenido del cuerpo. Elemento Figure

La etiqueta `<figure>` ha sido creada para ayudarnos a ser más específicos en la declaración del contenido del documento. Antes de que se introdujera este elemento, no podíamos identificar el contenido que era parte de la información pero independiente a la vez como por ejemplo las imágenes, ilustraciones, videos, etc.

Normalmente estos elementos son parte del contenido principal pero se pueden mover sin perjudicar al flujo del documento. Cuando existe este tipo de información, hay que utilizar las etiquetas `<figure>` para identificarlas.

`<figure>`

```
<!DOCTYPE html>
<html lang="es">
<head>
<meta charset="utf-8">
<meta name="description" content="Ejemplo de HTML5">
<meta name="keywords" content="HTML5">
<title>edu4java Home</title>
<link rel="stylesheet" href="edu4javastyles.css">
</head>
<body>
<header>
<h1>edu4java</h1>
</header>
<nav>
<ul>
<li>Tutoriales Web-Html</li>
<li>Tutoriales Juegos Android</li>
<li>Tutoriales Java para principiantes</li>
<li>Tutoriales Sql</li>
```

```
</li>Contacto</li>
</ul>
</nav>
<section>
<article>
<header>
<hgroup>
<h1>VIDEO TUTORIALES JAVA </h1>
<h3>Primer programa, con Eclipse en Español. Como programar en java </h3>
</hgroup>
</header>
<figure>
<iframe src="http://www.youtube.com/embed/Y5QI2IAoIjw"></iframe>
<figcaption>
Este es el video del primer tutorial de Java para principiantes.
</figcaption>
</figure>
<footer>
<p>comentarios (0)</p>
</footer>
</article>
</section>
<aside >
English Version
</aside>
<footer>
Copyright &copy; 2012-2013
</footer>
</body>
</html>
```

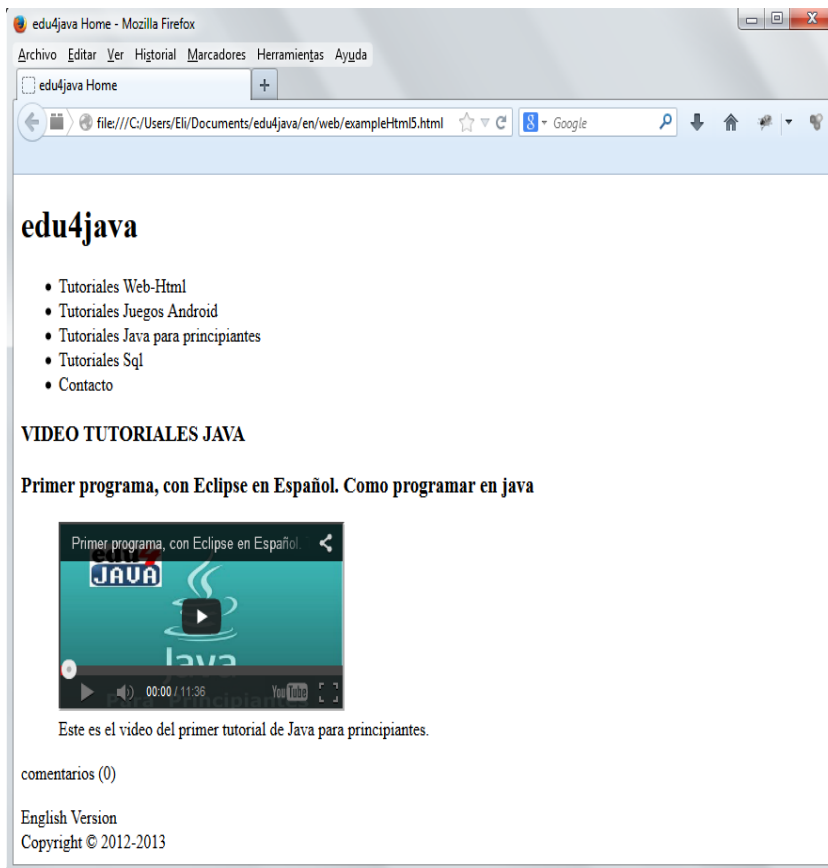
En el ejemplo, hemos insertado un video (<iframe src="http://www.youtube.com/embed/Y5QI2IAoIjw"></iframe>), después del texto. Esto se hace frecuentemente; comúnmente el texto se enriquece con imágenes o videos.

Las etiquetas <figure> nos permite encapsular estos complementos visuales y diferenciarlos así del resto del contenido.

En el código que vemos en el ejemplo, podemos ver un elemento extra dentro de <figure>. Comúnmente las unidades de información como imágenes o videos se describen con un texto corto debajo. HTML5 nos provee con un elemento para colocar e identificar esta leyenda.

Las etiquetas <figcaption> encapsulan la leyenda relacionada a la <figure> y establece una relación entre los dos elementos y el contenido.

Este es el resultado:



TABLAS en HTML5

En el pasado las tablas de HTML se utilizaban para controlar el layout de los sites (distribución de objetos en la página), de forma de mantener las proporciones del contenido, crear secciones, dividir la información, etc. Actualmente esto es una mala práctica, de hecho con las nuevas especificaciones del elemento **table** en HTML5 se ha hecho lo posible para que este sea utilizado para su verdadero propósito que es mostrar datos en arreglos de 2 dimensiones.

Elemento table

El elemento **table** es el gran contenedor y puede estar dentro de cualquier elemento que sea de flujo, como por ejemplo los **div**. Dentro del elemento **table** tendremos los elementos: **caption**, **colgroup**, **thead**, **tbody**, **tfoot**, **tr**, **th** y **td**, los cuales son los requeridos para darle la constitución a nuestra tabla, adicionalmente el nuevo estándar ha hecho que los siguientes atributos queden obsoletos dentro de **table** y tengan que utilizarse exclusivamente por CSS: *summary*, *align*, *width*, *bgcolor*, *cellpadding*, *cellspacing*, *frame*, *rules*.

Elemento tr

Como bien sabemos las tablas se componen de dos elementos básicos, filas y columnas, donde las filas representan cada registro de información y las columnas la estructura y el tipo de la información. El elemento utilizado para crear las filas es el **tr**. Este puede ser hijo de los siguientes elementos: **table**, **thead**, **tfoot**, **tbody**, como vemos puede ser hijo directo de una tabla o hijo de los elementos que son hijos de la tabla y que definen su estructura. Debe contener 1 o más elementos **td** o **th** según corresponda, en el nuevo estándar de HTML5 igualmente se ha colocado alguno de sus atributos obsoletos, estos son: *align*, *char*, *charoff*, *valign*, *bgcolor*, los cuales han de trabajarse exclusivamente por CSS.

Elemento td

Este elemento define las columnas dentro de nuestras filas en la tabla, lo que nos permite separar la información dentro de un registro y así poder clasificarla según nuestras necesidades, es hijo del elemento **tr**, los atributos validos que puede poseer son: *colspan*, *rowspan*, *headers* y los atributos obsoletos que ahora deben manejarse en el CSS son: *abbr*, *axis*, *align*, *width*, *char*, *charoff*, *valign*, *bgcolor*, *height*, *nowrap*, *scope*.

Nuestra primera tabla

Pongamos en práctica lo que hemos visto hasta ahora, ya sabemos que para nuestra tabla necesitamos filas y columnas dentro de nuestro elemento contenedor o padre, veamos a continuación como lo llevamos a código HTML:

```
1.  <!DOCTYPE HTML>
2.  <html>
3.      <head>
4.          <title>Ejemplo</title>
5.      </head>
6.      <body>
7.          <table>
8.              <tr>
9.                  <td>Apples</td>
10.                 <td>Green</td>
11.                 <td>Medium</td>
12.             </tr>
13.             <tr>
14.                 <td>Oranges</td>
15.                 <td>Orange</td>
16.                 <td>Large</td>
17.             </tr>
18.         </table>
19.     </body>
20. </html>
```

Este código, nos da la estructura de una tabla de 2 filas con 3 columnas, pero como se podrá observar, es texto simple. Ahora la haremos un poco más avanzada con contenido CSS (más adelante veremos en detalle el concepto y uso de CSS y en particular CSS3).

Elemento th

El elemento **th** nos permite crear encabezados a nuestra tabla de forma de poder identificar visualmente nuestras columnas al igual que **td** es hijo de **tr**. Sus atributos son: **colspan**, **rowspan**, **scope**, **headers**, los atributos obsoletos en esta especificación de HTML5 son los siguientes: **abbr**, **axis**, **align**, **width**, **char**, **charoff**, **valign**, **bgcolor**, **height**, and **nowrap**, **scope**, si nos fijamos en detalle este elemento funciona muy similar a **td**.

Ahora veamos cómo utilizarlo para construir una tabla con un poco más de formato de lo que habíamos hecho anteriormente:

```
1.  <!DOCTYPE HTML>
2.  <html>
3.      <head>
4.          <title>Ejemplo</title>
5.      </head>
6.      <body>
7.          <table>
8.              <tr>
9.                  <th>Rank</th><th>Name</th>
10.                 <th>Color</th><th>Size</th>
```

```
11.         </tr>
12.         <tr>
13.             <th>Favorite:</th>
14.             <td>Apples</td><td>Green</td><td>Medium</td>
15.         </tr>
16.         <tr>
17.             <th>2nd Favorite:</th>
18.             <td>Oranges</td><td>Orange</td><td>Large</td>
19.         </tr>
20.         <tr>
21.             <th>3rd Favorite:</th>
22.             <td>Pomegranate</td><td>A kind of greeny-
red</td>
23.             <td>Varies from medium to large</td>
24.         </tr>
25.     </table>
26. </body>
27. </html>
```

Como pudimos visualizar en este ejemplo podemos colocar el elemento **th** dentro de los **tr**, su función es similar al **td**, solo que este se utiliza para generar cabeceras, el código mostrado nos da como resultado la siguiente tabla:



Rank	Name	Color	Size
Favorite:	Apples	Green	Medium
2nd Favorite:	Oranges	Orange	Large
3rd Favorite:	Pomegranate	A kind of greeny-red	Varies from medium to large

Nuestra tabla empieza a lucir mucho mejor. Sin embargo aún es muy básica, y la información no está tan clara como debería quedar, por ejemplo las cabeceras y el contenido no se corresponden de buena manera, y visualmente es difícil distinguir que pertenece a que, para ello vamos a utilizar CSS y veremos cómo resolveremos en gran parte esta situación. Para ello utilizaremos lo siguiente:

```
1.     <style>
2.         tr > th { text-align:left; background:grey;
color:white}
```



```
3.         tr > th:only-of-type {text-align:right; background:
    lightgrey; color:grey}
4.         </style>
5.
```

Tenemos dos condiciones para los **th**, ambas empiezan diciéndonos que son hijos de **tr**, sin embargo la primera le colocamos que su alineación será a la izquierda, tendrá letras blancas y su fondo será gris, en la segunda indicamos que cuando solo exista un **th** por cada **tr** lo alinearemos a la derecha, colocaremos un gris más claro de fondo y las letras será gris más oscuro. Veamos el código resultante:

```
1.  <!DOCTYPE HTML>
2.  <html>
3.    <head>
4.      <title>Example</title>
5.      <style>
6.        tr > th {text-align:left; background:grey;
    color:white}
7.        tr > th:only-of-type {text-align:right; background:
    lightgrey; color:grey}
8.      </style>
9.    </head>
10.   <body>
11.     <table>
12.       <tr>
13.         <th>Rank</th><th>Name</th><th>Color</th><th>Size</th>
14.       </tr>
15.       <tr>
16.         <th>Favorite:</th><td>Apples</td><td>Green</td><td>Medium</td>
17.       </tr>
18.       <tr>
19.         <th>2nd
    Favorite:</th><td>Oranges</td><td>Orange</td><td>Large</td>
20.       </tr>
21.       <tr>
22.         <th>3rd
    Favorite:</th><td>Pomegranate</td><td>A kind of greeny-red</td>
23.         <td>Varies from medium to large</td>
24.       </tr>
25.     </table>
26.   </body>
27. </html>
```

Ahora veamos el resultado de este código y notaremos como de esta forma nuestra tabla es mucho más organizada y podemos distinguir a que columna pertenece cada dato.



The screenshot shows a web browser window titled 'Example' with the address 'titan/listings/example.html'. It displays a table with the following content:

Rank	Name	Color	Size
Favorite:	Apples	Green	Medium
2nd Favorite:	Oranges	Orange	Large
3rd Favorite:	Pomegranate A kind of greeny-red Varies from medium to large		

A 'solvetic.com' watermark is visible in the bottom right corner of the browser window.

Como podemos ver, este ejemplo nos da una visión de lo que una tabla debería ser, sin embargo aún dista de ser perfecta. ¿Qué pasaría por ejemplo, si agregamos otros **th**, en la fila de la información? Perderíamos el formato, lo que nos tendría trabajando cada vez que la tabla cambie de estructura. Para evitar esto existen 3 elementos que nos dividen lógicamente la tabla por decirlo de alguna forma, estos son: **thead**, **tbody**, **tfoot**. Ya con lo que hemos visto estamos en capacidad de entender sin mucha explicación teórica, así que pasemos a un ejemplo práctico de esto.

```
1.  <!DOCTYPE HTML>
2.  <html>
3.      <head>
4.          <title>Example</title>
5.          <style>
6.              thead th, tfoot th { text-align:left; background:grey;
color:white}
7.              tbody th { text-align:right; background: lightgrey;
color:grey}
8.          </style>
9.      </head>
10.     <body>
11.         <table>
12.             <thead>
13.                 <tr>
14.                     <th>Rank</th><th>Name</th><th>Color</th><th>Size</th>
15.                 </tr>
16.             </thead>
17.             <tfoot>
18.                 <tr>
19.                     <th>Rank</th><th>Name</th><th>Color</th><th>Size</th>
```

```
20.         </tr>
21.         </tfoot>
22.         <tbody>
23.         <tr>
24.         <th>Favorite:</th><td>Apples</td><td>Green</td><td>Medium</td>
25.         </tr>
26.         <tr>
27.         <th>2nd
Favorite:</th><td>Oranges</td><td>Orange</td><td>Large</td>
28.         </tr>
29.         <tr>
30.         <th>3rd
Favorite:</th><td>Pomegranate</td>
31.         <td>A kind of greeny-
red</td><td>Varies from medium to large</td>
32.         </tr>
33.         </tbody>
34.     </table>
35. </body>
36. </html>
```

Ahora veamos el resultado de esto:



Como vemos es en esencia lo primero que habíamos hecho, ahora en el pie de la tabla vemos que tenemos la cabecera por igual, sin embargo en el código vimos como son dos estructuras diferentes, al igual que el contenido. Con esto, finalizamos el tema de tablas en HTML5, sólo nos queda realizar unos cuantos ejercicios y practicar lo aprendido.

HTML5 – Formularios

La web 2.0 se centra en el usuario. Y por ello hay que hacer las interfaces más intuitivas, más naturales, más prácticas y más estéticas.

Los formularios son las interfaces más importantes; permiten al usuario insertar datos, tomar decisiones, comunicar información y cambiar el comportamiento de una aplicación.

Durante los últimos años se han creado códigos adaptados y librerías para procesar los formularios desde los ordenadores de los usuarios. HTML5 hace que estas propiedades sean estándar, aportando nuevos atributos, elementos y un API completo. (API: *Application Programming Interface*, es una función o proceso ya programado con un propósito, funcionalidad y condiciones específicas).

Actualmente la capacidad para procesar la información en los formularios en tiempo real se ha incorporado a los navegadores y está completamente estandarizado.

En HTML5, estos nuevos tipos, no sólo están especificando que tipo de dato se espera, sino que además le está diciendo al navegador que hacer con la información recibida. El navegador procesará los datos del input teniendo en cuenta el valor del atributo y validará o no la entrada de la información.

Formularios básicos

Los formularios más sencillos se pueden crear utilizando solamente dos etiquetas: `<form>` y `<input>`. Si se considera el formulario que muestra la siguiente imagen:



Formulario sencillo definido con las etiquetas form e input

El código HTML necesario para definir el formulario anterior se muestra a continuación:

```
<html>
```

```
<head><title>Ejemplo de formulario sencillo</title></head>
<body>
  <h3>Formulario muy sencillo</h3>
  <form name="miform" id="miform" method="get">
    Escribe tu nombre:&nbsp;   <input type="text" name="nombre" value="" />
    <br></br>
    <input type="submit" value="Enviar" />
  </form>
</body>
</html>
```

La etiqueta `<form>` encierra todos los contenidos del formulario (botones, cuadros de texto, listas desplegables) y la etiqueta `<input>` permite definir varios tipos diferentes de elementos (botones y cuadros de texto).

Etiqueta	<code><form></code>
Atributos propios	<ul style="list-style-type: none">• <code>action = "url"</code> - Indica la URL que se encarga de procesar los datos del formulario• <code>method = "POST o GET"</code> - Método HTTP empleado al enviar el formulario• <code>enctype = "application/x-www-form-urlencoded o multipart/form-data"</code> - Tipo de codificación empleada al enviar el formulario al servidor (sólo se indica de forma explícita en los formularios que permiten adjuntar archivos)• <code>accept = "tipo_de_contenido"</code> - Lista separada por comas de todos los tipos de archivos aceptados por el servidor (sólo para los formularios que permiten adjuntar archivos)• Otros: <code>accept-charset</code>, <code>onsubmit</code>, <code>onreset</code>
Descripción	Se emplea para insertar un formulario en la página

La mayoría de formularios utilizan sólo los atributos `action` y `method`. El atributo `action` indica la URL de la aplicación del servidor que se encarga de procesar los datos introducidos por los usuarios. Esta aplicación también se encarga de generar la respuesta que muestra el navegador. El atributo `method` establece la forma en la que se envían los datos del formulario al servidor. Este atributo hace referencia al método HTTP, por lo que no es algo propio de HTML. Los dos valores que se utilizan en los formularios son GET y POST. De esta forma, casi todos los formularios incluyen el atributo `method="get"` o el atributo `method="post"`. Al margen de otras diferencias técnicas, el método POST permite el envío de mucha más información que el método GET. En general, el método GET admite como máximo el envío de unos 500 bytes de información. La otra gran limitación del método GET es que no permite el envío de archivos adjuntos con el formulario. Además, los datos enviados mediante GET se ven en la barra de direcciones del navegador (se añaden al final de la URL de la página), mientras que los datos enviados mediante POST no se pueden ver tan fácilmente. Si no sabes que método elegir

para un formulario, existe una regla general que dice que el método GET se debe utilizar en los formularios que no modifican la información (por ejemplo en un formulario de búsqueda). Por su parte, el método POST se debería utilizar cuando el formulario modifica la información original (insertar, modificar o borrar alguna información). El ejemplo más común de formulario con método GET es el de los buscadores. Si realizas una búsqueda con tu buscador favorito, verás que las palabras que has introducido en tu búsqueda aparecen como parte de la URL de la página de resultados. Del resto de atributos de la etiqueta `<form>`, el único que se utiliza ocasionalmente es `enctype`. Si bien esto está fuera del alcance de la materia, podemos decir que este atributo es imprescindible en los formularios que permiten adjuntar archivos.

Elementos de Formulario

Los elementos de formulario como botones y cuadros de texto también se denominan "*campos de formulario*" y "*controles de formulario*". La mayoría de controles se crean con la etiqueta `<input>`, por lo que su definición formal y su lista de atributos es muy extensa:

Etiqueta	<code><input></code>
Atributos propios	<ul style="list-style-type: none">• <code>type = "text password checkbox radio submit reset file hidden image button"</code> - Indica el tipo de control que se incluye en el formulario• <code>name = "texto"</code> - Asigna un nombre al control (es imprescindible para que el servidor pueda procesar el formulario)• <code>value = "texto"</code> - Valor inicial del control• <code>size = "unidad_de_medida"</code> - Tamaño inicial del control (para los campos de texto y de password se refiere al número de caracteres, en el resto de controles se refiere a su tamaño en píxel)• <code>maxlength = "numero"</code> - Máximo número de caracteres para los controles de texto y de password• <code>checked = "checked"</code> - Para los controles checkbox y radiobutton permite indicar qué opción aparece preseleccionada• <code>disabled = "disabled"</code> - El control aparece deshabilitado y su valor no se envía al servidor junto con el resto de datos• <code>readonly = "readonly"</code> - El contenido del control no se puede modificar• <code>src = "url"</code> - Para el control que permite crear botones con imágenes, indica la URL de la imagen que se emplea como botón de formulario• <code>alt = "texto"</code> - Descripción del control
Descripción	Se emplean para insertar un control en un formulario

A continuación se muestran ejemplos de los controles que se pueden crear con la etiqueta `<input>`.

Cuadro de texto

Se trata del elemento más utilizado en los formularios. En el caso más sencillo, se muestra un cuadro de texto vacío en el que el usuario puede escribir cualquier texto:

Nombre

Ejemplo de etiqueta input (type=text)

A continuación se muestra el código HTML correspondiente al ejemplo anterior:

```
Nombre <br/>  
<input type="text" name="nombre" value="" />
```

El atributo `type` diferencia a cada uno de los diez controles que se pueden crear con la etiqueta `<input>`. Para los cuadros de texto, su valor es `text`. El atributo `name` es el más importante en los campos del formulario. De hecho, si un campo no incluye el atributo `name`, sus datos no se envían al servidor. El valor que se indica en el atributo `name` es el nombre que utiliza la aplicación del servidor para obtener el valor de este campo de formulario.

Cuando el usuario pulsa el botón de envío del formulario, el navegador envía los datos a una aplicación del servidor para que procese la información y genere una respuesta adecuada. En el servidor, la aplicación que procesa los datos debe obtener en primer lugar toda la información introducida por el usuario. Para ello, utiliza el valor del atributo `name` para obtener los datos de cada control del formulario.

Como el valor del atributo `name` se utiliza en aplicaciones programadas, es esencial ponerse de acuerdo con el programador de la aplicación, no se debe modificar su valor sin modificar la aplicación y no se deben utilizar caracteres problemáticos en programación (espacios en blanco, acentos y caracteres como ñ o ç).

El atributo `value` se emplea para establecer el valor inicial del cuadro de texto. Si se crea un formulario para insertar datos, los cuadros de texto deberían estar vacíos. Por lo tanto, o no se añade el atributo **value** o se incluye con un valor vacío `value=""`. Si por el contrario se crea un formulario para modificar datos, lo lógico es que se muestren inicialmente los datos guardados en el sistema. En este caso, el atributo `value` incluirá el valor que se desea mostrar: `<input type="text" name="nombre" value="Juan Pérez" />`

Si no se especifica un tamaño, el navegador muestra el cuadro de texto con un tamaño predeterminado. El atributo **size** permite establecer el tamaño, en caracteres, con el que se muestra el cuadro de texto. Su uso es imprescindible en muchos formularios, en los que algunos campos como la dirección deben mostrar más caracteres de lo normal (`<input size="100" ...>`) y otros campos como el código postal deben mostrar menos caracteres de lo normal (`<input size="5" ...>`).

Además de controlar el tamaño con el que se muestra un cuadro de texto, también se puede limitar el tamaño del texto introducido. El atributo `maxlength` permite establecer el máximo número de caracteres que el usuario puede introducir en un cuadro de texto. Su uso es imprescindible para campos como el código postal, el número de la Seguridad Social y cualquier otro dato con formato predefinido y limitado.

Por último, el atributo **`readonly`** permite que el usuario pueda ver los contenidos del cuadro de texto pero no pueda modificarlos y el atributo **`disabled`** deshabilita un cuadro de texto de forma que el usuario no pueda modificarlo y además, el navegador no envía sus datos al servidor.

TextArea

Este control es similar al control `TEXT`, salvo que permite el ingreso de muchas líneas de texto.

La marca `TEXTAREA` en HTML tiene dos propiedades: `rows` y `cols` que nos permiten indicar la cantidad de filas y columnas a mostrar en pantalla.

```
<textarea namd="carta" rows="10" cols="50" ></textarea>
```

Cuadro de contraseña

La única diferencia entre este control y el cuadro de texto normal es que el texto que el usuario escribe en un cuadro de contraseña no se ve en la pantalla. En su lugar, los navegadores ocultan el texto utilizando asteriscos o círculos, por lo que es ideal para escribir contraseñas y otros datos sensibles.

Contraseña

Ejemplo de etiqueta `input` (`type=password`)

```
Contraseña <br/>  
<input type="password" name="contrasena" value="" />
```

Cambiando el valor del atributo `type` por `password` se transforma el cuadro de texto normal en un cuadro de contraseña. Todos los demás atributos se utilizan de la misma forma y tienen el mismo significado.

Checkbox

Los checkbox o "*casillas de verificación*" son controles de formulario que permiten al usuario seleccionar y deseleccionar opciones individualmente. Aunque en ocasiones se muestran varios checkbox juntos, cada uno de ellos es completamente independiente del resto. Por este motivo, se utilizan cuando el usuario puede activar y desactivar varias opciones relacionadas pero no

excluyentes.

Puestos de trabajo buscados

- ☐ Dirección
- ☐ Técnico
- ☐ Empleado

Ejemplo de etiqueta input (type=checkbox)

```
Puestos de trabajo buscados <br></br>
<input name="puesto_directivo" type="checkbox" value="direccion"/> Dirección
<input name="puesto_tecnico" type="checkbox" value="tecnico"/> Técnico
<input name="puesto_empleado" type="checkbox" value="empleado"/> Empleado
```

El valor del atributo type para estos controles de formulario es checkbox. Como se muestra en el ejemplo anterior, el texto que se encuentra al lado de cada *checkbox* no se puede establecer mediante ningún atributo, por lo que es necesario añadirlo manualmente fuera del control del formulario. Si no se añade un texto al lado de la etiqueta `<input />` del *checkbox*, el usuario sólo ve un pequeño cuadrado sin ninguna información relativa a la finalidad de ese *checkbox*.

El valor del atributo value, junto con el valor del atributo name, es la información que llega al servidor cuando el usuario envía el formulario.

Si se quiere mostrar un *checkbox* seleccionado por defecto, se utiliza el atributo checked. Si el valor del atributo es checked, el *checkbox* se muestra seleccionado. En cualquier otro caso, el *checkbox* permanece sin seleccionar. Aunque resulta redundante que el nombre y el valor del atributo sean idénticos, es obligatorio indicarlo de esta forma porque los atributos en HTML no pueden tener valores vacíos:

```
<input type="checkbox" checked="checked" ... /> Checkbox seleccionado por defecto
```

Radiobutton

Los controles de tipo radiobutton son similares a los controles de tipo checkbox, pero presentan una diferencia muy importante: son mutuamente excluyentes. Los radiobutton se utilizan cuando el usuario solamente puede escoger una opción entre las distintas opciones relacionadas que se le presentan. Cada vez que se selecciona una opción, automáticamente se deselecta la otra opción que estaba seleccionada.

Sexo

- ☒ Hombre
- ☐ Mujer

Ejemplo de etiqueta input (type=radio)

```
Sexo <br/>
```

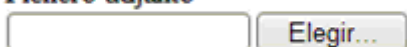
```
<input type="radio" name="sexo" value="hombre" checked="checked" /> Hombre  
<input type="radio" name="sexo" value="mujer" /> Mujer
```

El valor del atributo `type` para estos controles de formulario es `radio`. El atributo `name` se emplea para indicar los `radiobutton` que están relacionados. Por lo tanto, cuando varios *radiobutton* tienen el mismo valor en su atributo `name`, el navegador sabe que están relacionados y puede deseleccionar una opción del grupo de *radiobutton* cuando se seleccione otra opción.

Archivos adjuntos

Los formularios también permiten adjuntar archivos para subirlos al servidor. Aunque desde el punto de vista de HTML y del navegador no existe ninguna limitación sobre el número, tipo o tamaño total de los archivos que se pueden adjuntar, todos los servidores añaden restricciones por motivos de seguridad.

Fichero adjunto

Un formulario de archivo adjunto que consiste en un campo de texto vacío y un botón etiquetado como "Elegir...".

Ejemplo de etiqueta input (type=file)

Fichero adjunto

```
<input type="file" name="adjunto" />
```

El valor del atributo `type` para este control de formulario es `file`. El navegador se encarga de mostrar un cuadro de texto donde aparece el nombre del archivo seleccionado y un botón que permite navegar por los directorios y archivos del ordenador del usuario.

Si se incluye un control para adjuntar archivos, es obligatorio añadir el atributo `enctype` en la etiqueta `<form>` del formulario. El valor del atributo `enctype` debe ser `multipart/form-data`, por lo que la etiqueta `<form>` de los formularios que permiten adjuntar archivos siempre es:

```
<form action="..." method="post" enctype="multipart/form-data">  
...  
</form>
```

Campos ocultos

Los campos ocultos se emplean para añadir información oculta en el formulario:

*Los campos ocultos
no se ven en pantalla

Ejemplo de etiqueta input (type=hidden)

```
<input type="hidden" name="url_previa" value="/articulo/primer.html" />
```

El valor del atributo type para este control de formulario es hidden. Los campos ocultos no se muestran por pantalla, de forma que el usuario desconoce que el formulario los incluye. Normalmente los campos ocultos se utilizan para incluir información que necesita el servidor pero que no es necesario o no es posible que la establezca el usuario.

Etiqueta <select>

La etiqueta <select> crea un menú desplegable. Cada opción ofrecida por dicho menú se representa por un elemento option.

Ejemplo:

```
<select>
  <option value="html">HTML</option>
  <option value="css">CSS</option>
  <option value="js">JavaScript</option>
  <option value="php">PHP</option>
</select>
```

Posibles Atributos de la Etiqueta <select>

Atributos	Valor	Descripción
disabled	disabled	Deshabilita el control para la entrada de datos.
multiple	multiple	Si está activado, permite selecciones múltiples.
name	nombre	Especifica un nombre para el menú de desplazamiento.
size	número	Especifica el número de filas de la lista que deberían ser visibles al mismo tiempo.

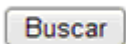
Atributos Estándar de la Etiqueta <select>

Atributos	Valor	Descripción
class	nombre de la clase	Asigna un nombre de clase. El atributo class actúa: <ul style="list-style-type: none">• Como selector para las hojas de estilo(CSS), cuando se asigna información de estilo a un conjunto de elementos.• Para procesos generales por parte del usuario.
id	nombre	Asigna un nombre a un elemento. El atributo id actúa: <ul style="list-style-type: none">• Como selector para las hojas de estilo(CSS).• Como vínculo destino para vínculos de hipertexto.• Como medio de hacer referencia a un elemento en particular desde un script.• Como nombre de un elemento object declarado.

		<ul style="list-style-type: none">• Para procesos generales por parte del usuario.
style	estilo	Este atributo especifica información de estilo para el elemento actual.
title	nombre	Este atributo ofrece información consultiva sobre el elemento para el cual se establece.
dir	ltr o rtl	Especifica la dirección del texto. Valores posibles: <ul style="list-style-type: none">• ltr : De izquierda a derecha (left to right).• rtl : De derecha a izquierda (right to left).
lang	código de lenguaje	Especifica el idioma base de los valores de los atributos y del texto contenido en un elemento. El atributo lang es útil para: <ul style="list-style-type: none">• Ayudar a los motores de búsqueda.• Ayudar a los sintetizadores de voz.• Ayudar al agente de usuario a hacer decisiones sobre separación de palabras, ligaduras, y espaciado.• Ayudar a los verificadores de ortografía y gramática.
tabindex	número	Especifica el orden de tabulación del elemento dentro del documento actual

Botón de envío de formulario

La mayoría de formularios dispone de un botón para enviar al servidor los datos introducidos por el usuario:



Ejemplo de etiqueta input (type=submit)

```
<input type="submit" name="buscar" value="Buscar" />
```

El valor del atributo type para este control de formulario es submit. El navegador se encarga de enviar automáticamente los datos cuando el usuario pincha sobre este tipo de botón. El valor del atributo value es el texto que muestra el botón. Si no se establece el atributo value, el navegador muestra el texto predefinido Enviar consulta.

Botón de reseteo del formulario

Aunque su uso era muy popular hace unos años, la mayoría de formularios modernos ya no utilizan este tipo de botón. Se trata de un botón especial que borra todos los datos introducidos por el usuario y devuelve el formulario a su estado original:

Borrar datos del formulario

Ejemplo de etiqueta input (type=reset)

```
<input type="reset" name="limpiar" value="Borrar datos del formulario" />
```

El valor del atributo type para este control de formulario es reset. Cuando el usuario pulsa este botón, el navegador borra toda la información introducida y muestra el formulario en su estado original. Si el formulario no contenía originalmente ningún valor, el botón de reset lo vuelve a mostrar vacío. Si el formulario contenía información, el botón reset vuelve a mostrar la misma información original.

Como es habitual en los botones de formulario, el atributo value permite establecer el texto que muestra el botón. Si no se utiliza este atributo, el navegador muestra el texto predefinido del botón, que en este caso es Restablecer.

Botón genérico

Algunos formularios complejos necesitan botones más avanzados que los de enviar datos (type="submit") y resetear el formulario (type="reset"). Por ese motivo, el estándar HTML/XHTML define un botón de tipo genérico:

Guardar Cambios

Ejemplo de etiqueta input (type=button)

```
<input type="button" name="guardar" value="Guardar Cambios" />
```

El valor del atributo type para este control de formulario es button. Si pruebas pulsar un botón de este tipo, verás que el navegador no hace nada: no envía los datos al servidor y no borra los datos introducidos. Este tipo de botones sólo son útiles si se utilizan junto con el lenguaje de programación JavaScript. Si la página incluye código JavaScript, los botones de este tipo se pueden programar para que realicen cualquier tarea compleja cuando se pulsa sobre ellos.

Botón de imagen

El aspecto de los botones de formulario se puede personalizar por completo, ya que incluso es posible utilizar una imagen como botón:



Ejemplo de etiqueta input (type=image)

```
<input type="image" name="enviar" src="accept.png" />
```

El valor del atributo `type` para este control de formulario es `image`. El atributo `src` indica la URL de la imagen que debe mostrar el navegador en lugar del botón normal.

Su principal ventaja es que permite personalizar por completo la estética de los botones y mostrarlos con un aspecto homogéneo en todos los navegadores. El principal inconveniente es que ralentiza la carga del formulario y que si se quiere modificar su aspecto, es necesario crear una nueva imagen.

Nuevos tipos de Input en HTML5

Tipo Email (correo electrónico)

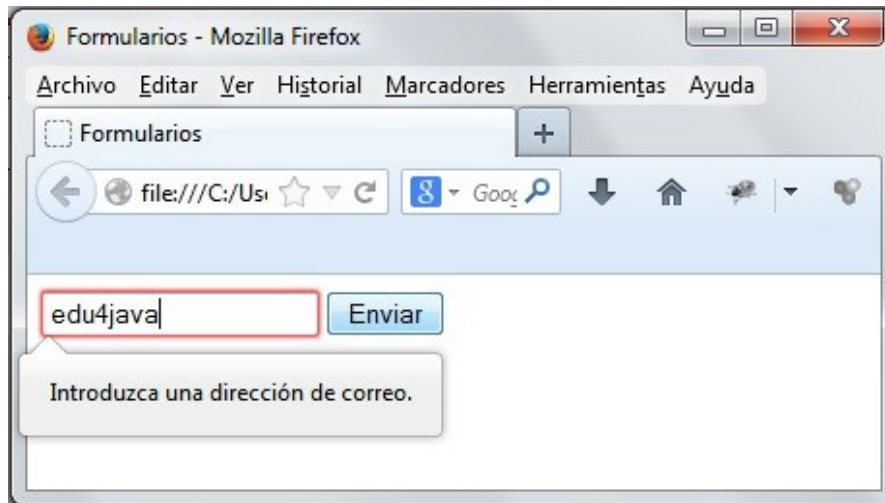
Casi todos los formularios del planeta tienen un campo *input* para insertar una dirección de correo electrónico. Hasta ahora, el único *tipo* disponible para este tipo de datos era el texto (*text*). El *type text* (tipo texto), representa un texto general, no un dato específico, por lo que había que controlar con Javascript si el texto insertado era una dirección de correo electrónica válida. Ahora es el navegador el que se encarga de validar la información insertada:

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get" autocomplete="on" >
<input type="email" name="micorreo" id="micorreo" placeholder="ingrese un
email" required autofocus />
<input type="submit" value="Enviar">
</form>
</section>
</body>
```

</html>

Este es el resultado utilizando el navegador Firefox:



El texto insertado en el campo es examinado por el navegador y validado o no como correo electrónico. En este caso, reconoce que no es una dirección correcta y salta un mensaje de error. Las respuestas de cada navegador no vienen determinadas por las especificaciones HTML5.

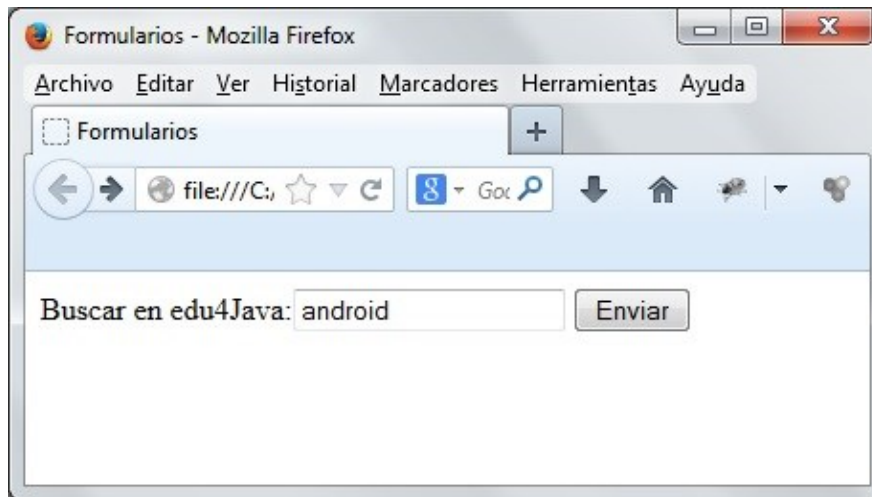
Tipo *Search* (Búsqueda)

El tipo *search* no controla la entrada de la información; simplemente es una indicación para los navegadores. Algunos navegadores cambian el diseño de este elemento, para señalar al usuario la función del campo.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Buscar en edu4Java:<input type="search" name="mibúsqueda" id="mibúsqueda">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Lo que obtenemos es un campo que se comporta como un campo de texto normal;



Tipo *URL*

Este tipo funciona exactamente igual que el *email*, pero para una dirección de una página web. Esta creado para recibir únicamente URLs absolutos y retornará un mensaje de error si el valor es inválido.

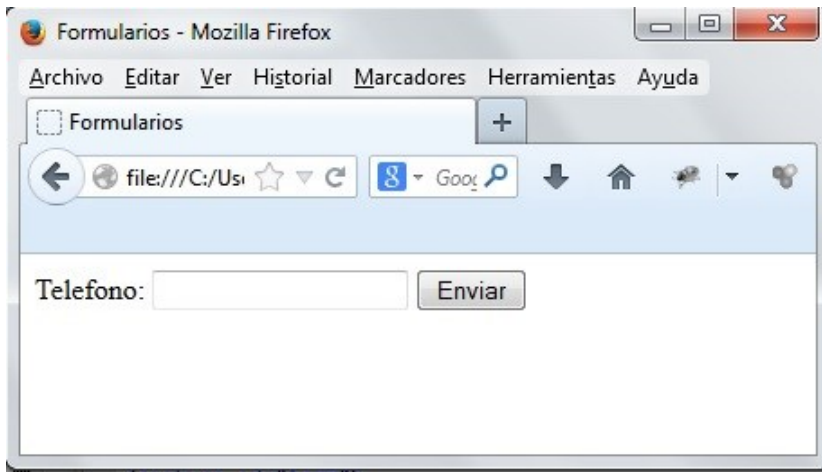
Tipo *Tel* (Teléfono)

Este tipo es para insertar números de teléfono. A diferencia de los tipos *email* y *url*, el tipo *tel*, no necesita una sintaxis específica. Es una indicación para el navegador, por si la aplicación necesita hacer ajustes teniendo en cuenta el dispositivo en el que se está ejecutando. Por ejemplo, un smartphone como el iPhone convierte su teclado a números de teléfono.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Telefono: <input type="tel" name="mitelfono" id="mitelfono">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este sería el resultado:



Tipo *Number*

Como indica su nombre, el tipo *number* sólo es válido cuando recibe un dato numérico.

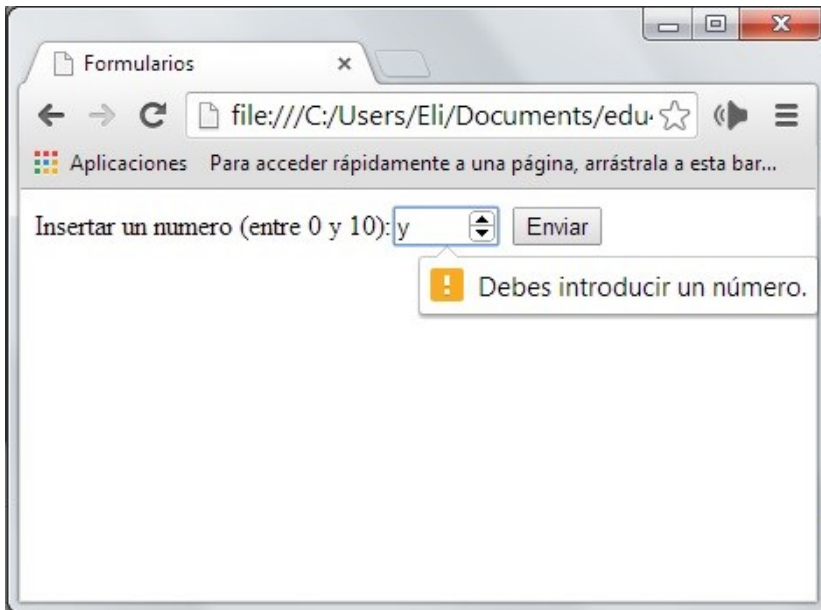
A continuación tenemos nuevos atributos que pueden ser útiles para este campo:

- *min*—El valor de este atributo determina el valor mínimo aceptado para este campo.
- *max*—El valor de este atributo determina el valor máximo aceptado para este campo.
- *step*—El valor de este atributo determina los intervalos en los que los valores del campo son incrementados y disminuidos. Por ejemplo, si se define un *step* de 5, con un mínimo de 0 y un máximo de 10, el navegador no permite especificar un valor entre 0 y 5 o entre 5 y 10.

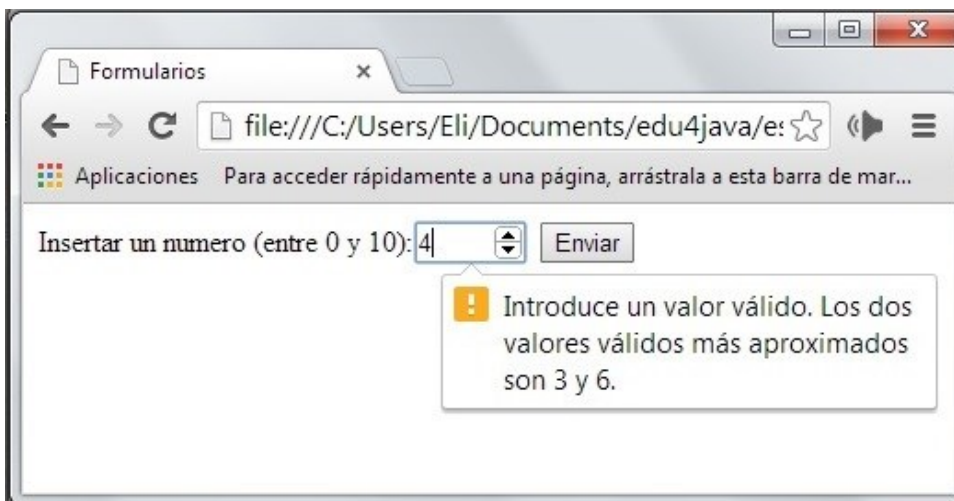
Ejemplo

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Insertar un número (entre 0 y 10):
<input type="number" name="minumber" id="minumber" min="0" max="10" step="3">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en el Navegador Chrome. Donde vemos que si introducimos una letra y no un número, nos salta un mensaje de error:



También nos salta un mensaje de error cuando insertamos un número que está fuera del intervalo de 3 que hemos definido con el atributo *step=3*:



Tipo *Range*

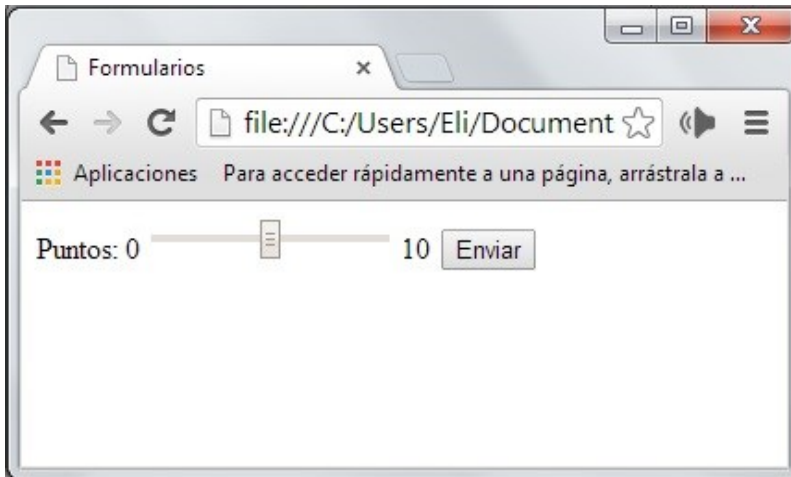
Este tipo hace que el navegador construya un nuevo control. Este control deja escoger al usuario el valor de entre un rango de números. Normalmente se muestra con un deslizador para mover los valores de arriba a abajo o con unas flechas, pero no existe un diseño estándar aún.

Este tipo utiliza los atributos *min* y *max* para definir los límites del rango. También puede tener el atributo *step* para definir los valores del intervalo en los que se incrementará o disminuirá el rango.

Ejemplo:

Puntos: `<input type="range" name="mirango" id="mirango" min="0" max="10" step="2">`

Este es el resultado:



Tipo *Date*

Este tipo también generará un control nuevo. En este caso, se incluyó para mejorar la inserción de fechas.

Un ejemplo es cuando el usuario está eligiendo la fecha de un vuelo, en este caso, el tipo *date* hará que el navegador nos muestre un calendario y que nosotros sólo tengamos que insertar el elemento `<input>` en el documento para que le aparezca al usuario.

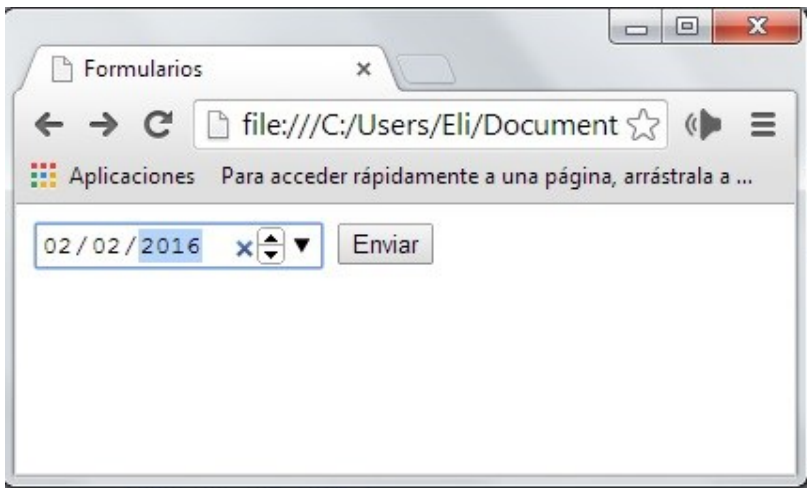
La interface no se declara en la especificación. Cada navegador suministra su propia interface y lo hará de acuerdo al dispositivo en el que ejecute la aplicación.

Normalmente, el valor esperado y generado tiene la sintaxis año/mes/día.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
<input type="date" name="mifecha" id="mifecha">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en Chrome:



Tipo *Week*

Este tipo suministra una interface similar a la de *date*, pero sólo para escoger semanas completas. Normalmente el valor esperado tiene la sintaxis 2014-W10, donde 2014 es el año y el 10 es el número de la semana.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Seleccione una semana:
<input type="week" name="misemana" id="misemana">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en Chrome:

Formularios

file:///C:/Users/Eli/Documents/edu4java/es/web/ejemplot

Aplicaciones Para acceder rápidamente a una página, arrástrala a esta barra de marcadores. Importar m...

Seleccione una semana: Semana 13, 2014 Enviar

marzo de 2014

Semana	lun	mar	mié	jue	vie	sáb	dom
9	24	25	26	27	28	1	2
10	3	4	5	6	7	8	9
11	10	11	12	13	14	15	16
12	17	18	19	20	21	22	23
13	24	25	26	27	28	29	30
14	31	1	2	3	4	5	6

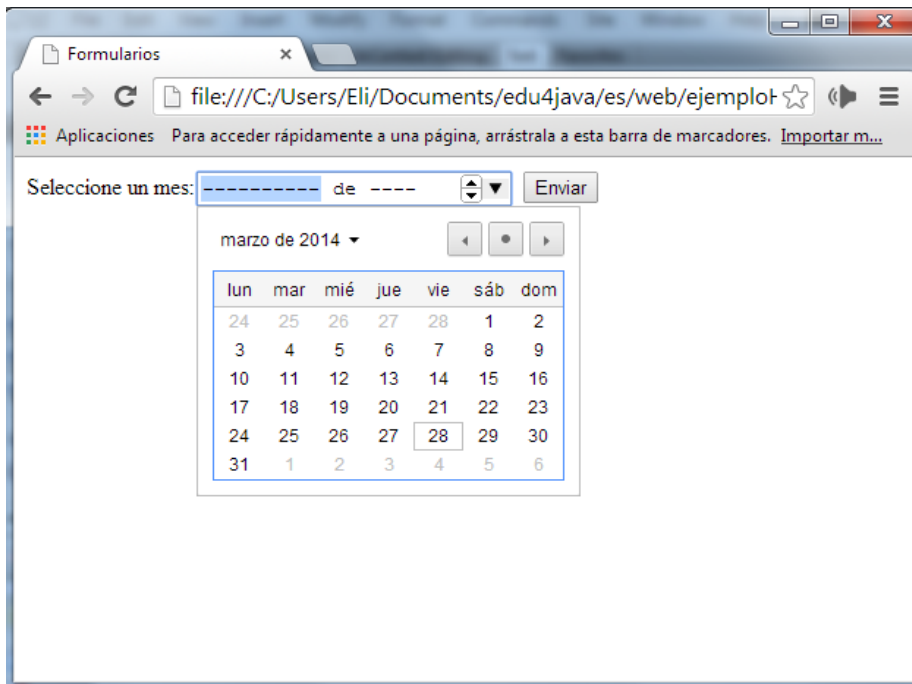
Tipo *Month*

Es similar al tipo anterior, aunque para un mes completo. El valor esperado tiene la sintaxis año-mes.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Seleccione un mes: <input type="month" name="mimes" id="mimes">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en Chrome:



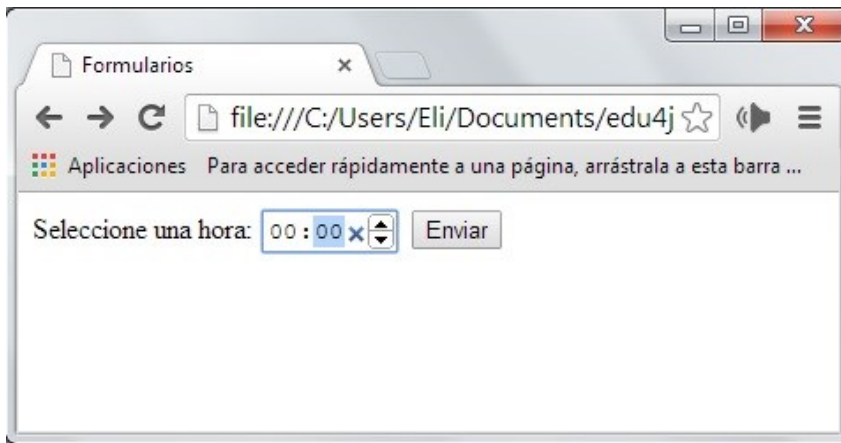
Tipo *Time*

El tipo *time* es similar a *date*, pero para fijar la hora. Normalmente tiene la sintaxis hora:minutos:segundos, pero también puede ser hora:minutos. Dependerá del navegador.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Seleccione una hora: <input type="time" name="mihora" id="mihora">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en Chrome:



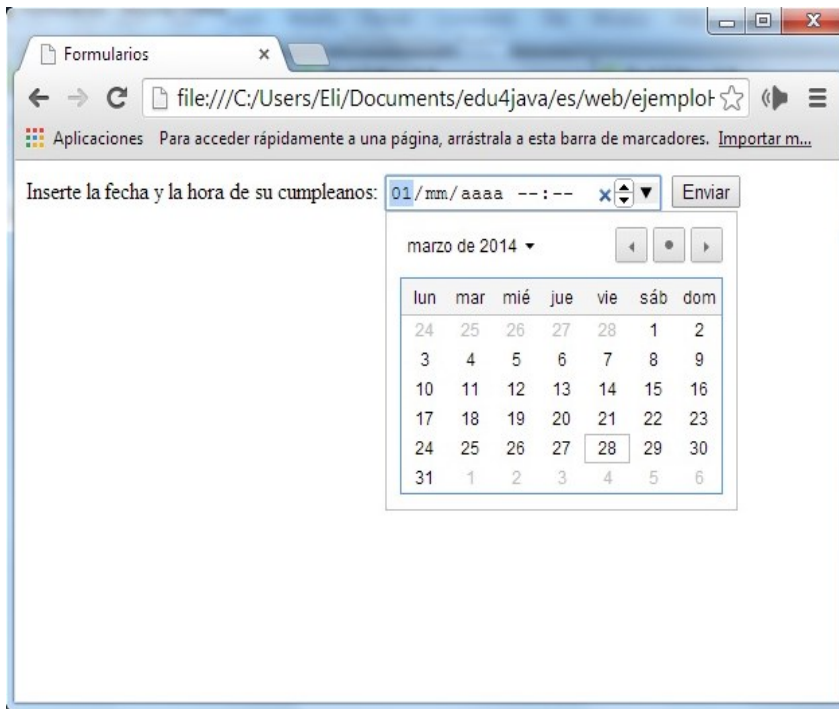
Tipo *Datetime-local*

El tipo *datetime-local* es un tipo de datetime (fecha y hora) sin zona horaria.

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Inserte la fecha y la hora de su cumpleaños:
<input type="datetime-local" name="mihorafechalocal" id="mihorafechalocal">
<input type="submit" value="Enviar">
</form>
</section>
</body>
</html>
```

Este es el resultado en el Navegador Chrome:



Tipo *Datetime*

El tipo *datetime* es para fecha y hora, incluyendo zona horaria.

```
<input type="datetime" name="mifechahora" id="mifechahora">
```

Tipo *Color*

Además de los tipos para las fechas y horas, también hay un tipo que nos suministra una interface predefinida para que el usuario pueda elegir un color. Normalmente, el valor esperado para este campo es un número hexadecimal, como #00FF00.

Ejemplo:

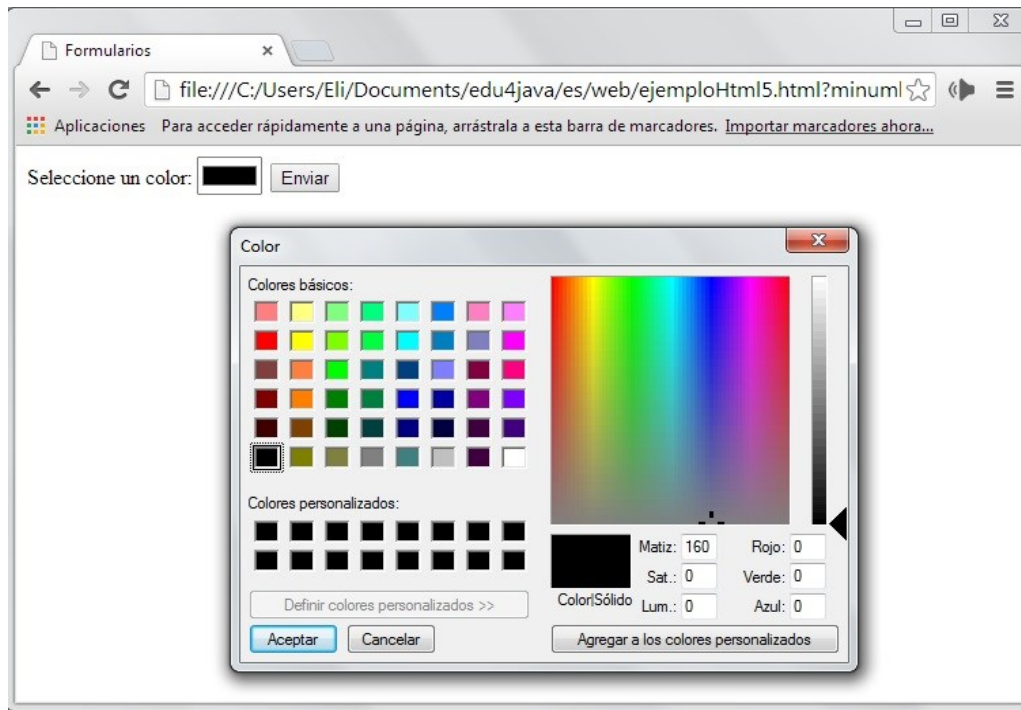
```
<!DOCTYPE html>
<html lang="es">
<head>
<title>Formularios</title>
</head>
<body>
<section id="form">
<form name="miform" id="miform" method="get">
Seleccione un color:
<input type="color" name="micolor" id="micolor">
<input type="submit" value="Enviar">
</form>
</section>
```



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

No existe una interface estándar para el color determinada por HTML5, pero los navegadores suelen mostrar una tabla de colores.

Este es el resultado en Chrome:



Uso del elemento DATALIST

Con la introducción de HTML5 al mundo de la programación, una gama de herramientas nuevas fueron proporcionadas para aumentar las posibilidades de lograr un mejor estructurado y semántico desarrollo web. Muchos de estos nuevos elementos fueron enfocados a la mejora del manejo de formularios, entre los más interesantes encontrados al elemento datalist.

La etiqueta datalist, es un concepto introducido por esta versión del lenguaje, para definir una lista de opciones que podemos presentar al usuario para su elección. Generalmente cuando realizamos un formulario, utilizamos el elemento select para delimitar las respuestas que puede dar un usuario, presentando una lista de opciones y de esa manera evitar que existan respuestas incorrectas. Cuando se presenta un caso donde necesitemos dar opciones pero sin limitar al usuario, es ahí donde entra datalist.

Si relacionamos la lista que definimos mediante datalist, con un elemento de tipo input text, podemos modificar su conducta por defecto y hacer que se despliegue un conjunto de sugerencias al momento de que el usuario ingrese algún carácter. El comportamiento que obtenemos, es algo que se puede observar en

muchos sitios de la actualidad; sin embargo, hasta hoy esto solo era posible de hacer utilizando JavaScript como complemento.

Uso de la etiqueta

Como ya dijimos el elemento `datalist` consiste en una lista, y como tal debemos definirla para poder utilizarla. Dentro de la etiqueta debemos establecer un conjunto de opciones, algo parecido a lo que hacemos con el elemento `select`, con la diferencia de que `datalist` no es un elemento que tenga control independiente, por ello se utiliza en complemento con un `input`.

Una definición sencilla, sería como la siguiente:

```
?  
1  
2  
3     <label>Cual es tu color favorito:</label>  
4     <input type="text" id="color_favorito" list="colores" />  
5     <datalist id="colores">  
6         <option value="Azul" />  
7         <option value="Amarillo" />  
8         <option value="Rojo" />  
9         <option value="Verde" />  
10        <option value="Negro" />  
11        <option value="Blanco" />  
12        <option value="Naranja" />  
13    </datalist>
```

Como se puede apreciar en el código, hacemos uso de un `input` sencillo de tipo texto, el cual cuenta con un **atributo que lleva por nombre "list"**. Este atributo es utilizado para indicarle al navegador lo que debe cargar al momento de que el usuario ingrese un valor en la caja del formulario. Para poder lograr esta relación, el valor del atributo `list` debe de coincidir con el `id` del elemento `datalist` que contiene la lista de valores o sugerencias para ese campo.

Después del elemento `input`, pasamos a definir la lista con el elemento `datalist`, esta estará **compuesta por un conjunto de elementos de tipo option**. Cada uno de estas opciones aparecerá cuando la letra que ingresemos en la caja coincida con su letra inicial. A medida que vayamos agregando más caracteres a la caja, la lista se ira refinando, hasta que quede la opción que más se asemeje a lo que se esta escribiendo.

Desgraciadamente el resultado puede variar de navegador a navegador, pero en términos generales obtendrás un resultado similar al que puedes observar en la página de Google.

Vale aclarar que el elemento `datalist`, puede ser utilizado con otros tipos de `input`, no solo texto. En base a lo que nos dice la W3C, esta clase de listas se puede utilizar con tipo **url, teléfono, color y fecha**.

Añadir más atributos

Podemos llevar las cosas a un nivel más adelante y utilizar el atributo `label` para complementar la estructura de las sugerencias. Sin embargo, tenemos que ser precavidos con el uso de este atributo, ya que el resultado que obtenemos luce distinto en los navegadores, hay incluso algunos navegadores que solo muestran el valor agregado en este atributo, como el caso de Firefox.

Si llegamos a utilizar los dos atributos, tanto `value` como `label`, debemos tener en mente que Opera filtrará en base a los dos atributos, Firefox utilizará el valor de `label` para desplegar la sugerencia y Chrome ignorará el valor de `label` para traer resultados.

```
1
2   <label>Dinos cual es tu color favorito:</label>
3   <input type="text" id="color_favorito" list="colores" />
4   <datalist id="colores">
5       <option value="Azul" label="como el cielo"/>
6       <option value="Amarillo" label="como el Sol"/>
7       <option value="Rojo" label="como la sangre"/>
8       <option value="Verde" label="como los campos"/>
9       <option value="Negro" label="como el carbón"/>
10      <option value="Blanco" label="como la pureza"/>
11      <option value="Naranja" label="como el cítrico"/>
12  </datalist>
```

Como hemos podido apreciar el elemento `datalist` aún se encuentra en camino a una estandarización, es por eso que por obvias razones no es soportado por todos los navegadores. Si estás dispuesto a dar soporte a navegadores viejos, olvídate por completo de utilizar esta etiqueta, o en su lugar ofrece una alternativa para el usuario cuando este ingrese desde una versión vieja.

Este elemento es soportado en la actualidad por casi todos los navegadores modernos en sus últimas versiones. Internet Explorer lo soporta desde su versión 10, Firefox desde la 4, Chrome y Opera también ofrecen soporte, sin embargo Safari aún no.

Resumen de Nuevos Elementos en HTML5

Para poder manejar mejor las necesidades actuales de internet, HTML5 ha incluido nuevos elementos para dibujar gráficos, mostrar imagen y sonido, para mejorar la estructura de las páginas y gestionar mejor los formularios, así como nuevos APIs para arrastrar y soltar, para encontrar tu posición geográfica, para guardar datos locales, etc.

A continuación tenemos una lista de los nuevos elementos HTML, introducidos por HTML5 y una descripción de para que se usan.

Nuevos elementos Semánticos/Estructurales

HTML5 ofrece nuevos elementos para una mejor estructura:

Etiqueta	Descripción
<code><header></code>	Define el header para un documento o una sección.
<code><hgroup></code>	Agrupar los elementos para el encabezado.
<code><nav></code>	Define los enlaces de navegación en un documento.
<code><section></code>	Define una sección en el documento.
<code><main></code>	Define el contenido principal del documento.
<code><article></code>	Define un artículo del documento.
<code><aside></code>	Define contenido secundario de la página.
<code><footer></code>	Define el pie del documento o sección.
<code><details></code>	Define detalles adicionales que el usuario puede ver o esconder.
<code><summary></code>	Define un encabezado visible para un elemento <code><details></code> .
<code><figure></code>	Define un contenido independiente, como una ilustración, diagrama, foto, etc.
<code><figcaption></code>	Define un título para un elemento <code><figure></code> .
<code><mark></code>	Define texto resaltado o marcado.
<code><time></code>	Define una fecha/hora.
<code><bdi></code>	Define una parte del texto que puede ser formateada de una manera diferente a la del texto principal.
<code><wbr></code>	Define un posible retorno de carro.
<code><dialog></code>	Define una caja de diálogo o ventana
<code><command></code>	Define un botón de comando al que el usuario puede llamar
<code><meter></code>	Define una medida escalar dentro de un rango establecido.

<progress>	Define el progreso de una tarea.
<ruby>	Define una anotación ruby (para tipografía del este asiático) .
<rt>	Define una pronunciación de caracteres (para tipografía del este asiático)
<rp>	Define lo que se puede mostrar en los navegadores que no soporta las anotaciones ruby.

Nuevos elementos Form

Etiqueta	Descripción
<datalist>	Define opciones predefinidas para una caja de texto input.
<keygen>	Define un campo generador de palabras clave(para envío de claves a través de formularios)
<output>	Define el resultado de un cálculo

El nuevo elemento <canvas>

Etiqueta	Descripción
<canvas>	Define dibujos gráficos utilizando JavaScript

Nuevos Elementos Media

Etiqueta	Descripción
<audio>	Define contenido de sonido o música
<video>	Define contenido de video o smovie
<source>	Define fuentes para <video> y <audio>
<track>	Defines rutas para <video> y <audio>
<embed>	Define containers para aplicaciones externas (como plug-ins)

Elementos eliminados

Los siguientes elementos de HTML 4.01 han sido eliminados de HTML5:

- <acronym>
- <applet>
- <basefont>
- <big>
- <center>

- <dir>
-
- <frame>
- <frameset>
- <noframes>
- <strike>
- <tt>

Ejemplos de mejoras e incorporaciones respecto a antiguas versiones

HTML5 se desarrolló para simplificar, especificar y organizar el código. Para lograr estos propósitos, se añadieron algunos atributos y etiquetas y se integró HTML con CSS y Javascript. Estas incorporaciones y mejoras de versiones anteriores afectan no sólo a nuevos elementos, sino también a como usamos los viejos.

Algunas etiquetas:

 debería ser utilizado para indicar énfasis (reemplazando la etiqueta <i> que utilizábamos antes).

Mi coche es rojo

 para mostrar importancia.

Mi coche es rojo

<mark> para resaltar texto importante de acuerdo con las circunstancias. En algunos buscadores, la palabra será resaltada con un fondo amarillo por defecto, pero se puede sobrescribir esos estilos con los propios utilizando CSS.

Mi <mark>coche</mark> es rojo

 debería ser utilizado solamente cuando ningún otro elemento sea apropiado

Mi coche es rojo

<small>

La nueva especificidad de HTML es evidente en algunos elementos como <small>. Este elemento, anteriormente, presentaba cualquier texto con una letra pequeña. La etiqueta se refería al tamaño del texto, independientemente de su significado.

En HTML5 la nueva función del elemento <small> es representar la "letra

pequeña", como derechos de copyright, de propiedad, etc.

```
<small>Copyright &copy; 2013 edu4Java</small>
```

<cite>

Otro elemento que ha cambiado su naturaleza para ser más específico es `<cite>`. Ahora las etiquetas `<cite>` engloban el título de un trabajo, como un libro, película, canción, etc.

```
<span>Me gusta la película <b>cite</b>Casablanca</b></span>
```

<address>

El elemento `<address>` es un elemento viejo que se ha convertido en un elemento estructural. Se usa para plasmar la información de contacto de, por ejemplo, el contenido de un elemento `<article>` o del cuerpo entero. Este elemento debería ser incluido dentro de un `<footer>`, como en el ejemplo:

```
<article>
<header>
<h1>Título del tutorial uno</h1>
</header>
Este es el texto del tutorial
<footer>
<b>address</b>
<a href="http://www.edu4java.com/index.html">edu4Java</a>
</b>address</b>
</footer>
</article>
```

Para finalizar, existe una serie de caracteres, que para expresarlos como contenido de una etiqueta html necesitan ser convertidos, de acuerdo a la siguiente tabla (en el caso de no especificar charset="utf-8" en el tag <meta>), a estos se los denomina **Entidad HTML**, y se utilizan para asegurarnos que el navegador los interprete correctamente:

Caracter	Entidad
¿	¿
?	?
Á	á
É	é
Í	í
Ó	ó
Ú	ú
Ü	ü
Ñ	ñ
“ (apertura)	“
” (cierre)	”
©	©
®	®
(espacio)	